International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-020-01221-4

ORIGINAL ARTICLE

Knowledge-driven graph similarity for text classification

Niloofer Shanavas[1] · **Hui Wang**[1] · **Zhiwei Lin**[1] · **Glenn Hawe**[1]

Received: 16 November 2019 / Accepted: 3 October 2020

Abstract
Automatic text classification using machine learning is significantly affected by the text representation model. The structural information in text is necessary for natural language understanding, which is usually ignored in vector-based representations. In this paper, we present a graph kernel-based text classification framework which utilises the structural information in text effectively through the weighting and enrichment of a graph-based representation. We introduce weighted co-occurrence graphs to represent text documents, which weight the terms and their dependencies based on their relevance to text classification. We propose a novel method to automatically enrich the weighted graphs using semantic knowledge in the form of a word similarity matrix. The similarity between enriched graphs, *knowledge-driven graph similarity*, is calculated using a graph kernel. The semantic knowledge in the enriched graphs ensures that the graph kernel goes beyond exact matching of terms and patterns to compute the semantic similarity of documents. In the experiments on sentiment classification and topic classification tasks, our knowledge-driven similarity measure significantly outperforms the baseline text similarity measures on five benchmark text classification datasets.

Keywords Automatic text classification · Document similarity measure · Graph-based text representation · Graph enrichment · Graph kernels · Supervised term weighting · SVM

1 Introduction

Research on automatic text classification has gained importance due to the information overload problem and the need for faster and more accurate extraction of knowledge from huge data sources. Text classification assigns predefined labels to documents based on their content. An important step in automatic text classification is the effective representation of text. Bag-of-words is the most commonly used text representation scheme and is based on term independence assumption, where a text document is regarded as a set of unordered terms and is represented as a vector. It is simple and fast, but ignores the structural information in text such as the syntactic and semantic information. In contrast, the *graph-based representation* scheme is much more expressive than the bag-of-words representation, and can represent structural information such as term dependencies. It has been shown that graph-based representation can outperform bag-of-words representation [12, 18, 26–28, 32, 37].

Document similarity is used in many text processing tasks such as text classification, clustering and information retrieval. Document similarity is usually measured as the distance/similarity between the vector representations of text documents under the assumption that terms are independent and unordered, thus the structural information in text is lost. Since the association between terms in text contributes towards the meaning of the text document, considering the structural information in measuring similarity can potentially improve the accuracy of document classification.

A graph kernel measures the similarity between graphs based on the comparison of graph substructures. Using a graph kernel to measure document similarity enables the consideration of structural information in text. The similarity value computed by a graph kernel is dependent on the information represented in the graphs. Therefore, the question of how to represent text using a graph is crucial to the graph kernel approach to similarity-based text classification. Two main challenges in this approach are (1) the effective representation of the structural information in text and (2) the efficient utilisation of the rich information in the graph

✉ Niloofer Shanavas
shanavas-n@ulster.ac.uk

[1] School of Computing, Ulster University, Jordanstown BT37 0QB, UK

Published online: 19 November 2020

representation to compute similarity based on the main content of the documents.

In this paper, we present a graph-based text classification framework addressing these challenges. The text document is initially represented by a weighted co-occurrence graph. Then it is transformed to an enriched document graph by automatically creating similar nodes and edges (or associations), using a similarity matrix based on word similarities. Since a supervised term weighting method is used to weight the terms and their associations, the matching terms and patterns contribute to document similarity based on their relevance. The graph enrichment enables the similarity measure to go beyond exact matching of terms and associations. We use an edge walk graph kernel to utilise the information in the enriched weighted graphs for calculating the similarity between text documents. The kernel function takes as input a pair of weighted co-occurrence graphs and gives as output a similarity value based on matching relevant content of the text documents. The kernel matrix is built by computing the similarity between every pair of text graphs, which is then used to train SVM, a kernel-based classifier, for learning and predicting the classes of documents. Our proposed text classification framework aims to represent text document more richly and utilise such rich information efficiently, therefore we can expect this approach to have improved performance, advancing the state-of-the-art in text classification. Hence, the novel contributions made in this paper are (1) the proposed weighting of the graph, (2) the automatic enrichment of graphs and (3) the application of the new graph-based text representation to build the knowledge-driven similarity measure.

The rest of the paper is organised as follows. Section 2 discusses related work. Section 3 introduces the proposed weighted graph representation of text documents. Section 4 presents the method for automatic graph enrichment using a knowledge base. Section 5 describes the utilisation of the information in the proposed graphs using graph kernels. Section 6 presents the experiments and results. Finally, Sect. 7 concludes the paper.

2 Related work

There are works on kernel methods [29] that allow us to compute the similarity between structured objects such as trees, graphs and sequences. Text can be viewed as structured objects and the kernels for structured objects can be applied to compare the text documents for different text processing tasks such as information retrieval, text classification and text clustering.

Graph kernels are instances of the R-convolution kernels [13] that provide a way for comparing discrete structures. R-convolution kernels compare objects by decomposing the objects into parts and combining the results of the comparisons of the parts of the objects. Different substructures such as random walks, shortest path, cycles, subtrees have been considered to compute the similarity between graphs. Gärtner et al. [10] defined the random walk graph kernel approach that counts all pairs of matching walks in the two graphs. Subtree kernels count the common subtree patterns in the graphs [25]. Kernels based on cyclic patterns consider common cycles in the graphs [14]. Borgwardt and Kriegel [6] defined the shortest path graph kernel that compares all the shortest paths in the graphs.

Lodhi et al. [17] proposed the idea of string kernels for measuring document similarity. A string kernel compares ordered subsequences of characters in the document which need not be contiguous. Similarly, Cancedda et al. [8] worked with word-sequence kernel that considers sequences of words instead of characters. The word-sequence kernels compute similarity based on the number of matching word sequences and non-contiguous subsequences are penalized.

The information in knowledge bases such as WordNet [19] and Wikipedia can be utilised to improve the performance of text classification. Siolas and d'Alché Buc [30] introduced semantic smoothing by incorporating a-priori knowledge from WordNet into text classification. The semantic smoothing of tf-idf feature vectors is performed using a smoothing matrix that contains the semantic similarity between words obtained using WordNet. This results in the increase in the feature value of the terms that are related semantically. Siolas et al. showed that the introduction of semantic knowledge in SVM and *k*-NN improves the classification performance. There are other works [5, 20] that used WordNet for designing a semantic smoothing kernel for text classification. They calculated the similarity between words based on the semantic relationship of these terms in WordNet. Cristianini et al. [9] incorporated into a kernel the semantic relations between terms calculated using LSI. Wang and Domeniconi [35] developed semantic kernels by embedding the knowledge derived from Wikipedia and used it to improve the performance of document classification.

Supervised semantic smoothing kernels exist that utilise class information in building a semantic matrix [1, 2, 36]. A sprinkled diffusion kernel that uses both co-occurrence information and class information for word sense disambiguation is presented in [36]. In this approach, the smoothing helps in increasing the semantic relationship between terms in the same class. But, it does not distinguish the common terms between classes. Class meaning kernel (CMK) [2] is a supervised semantic kernel that considers the meaningfulness of terms in the classes using Helmholtz principle from Gestalt theory. In order to increase the importance of class specific terms compared to common terms, the semantic smoothing is done using the semantic matrix built from class-based meaning values of terms. Class weighting kernel

(CWK) [1] smooths the representation of documents using class-based term weights that calculates the importance of the terms in the classes. Hence, there are different variants of semantic kernels with variations in the design of the semantic smoothing matrix. Since a document is represented as a vector and is based on a term independence assumption, these semantic kernels [1, 2, 5, 9, 20, 30, 35, 36] do not consider term dependencies such as the order of words or the distance between words in the computation of similarity between documents.

Walk-based kernels that are products of node kernels have been proposed that captures semantic similarity between words using word embeddings. Srivastava et al. [31] developed an approach that considers both syntactic and semantic similarity through a random walk-based kernel. It extends beyond label matching as word embeddings (SENNA) are used to represent words. Kim et al. [15] proposed a convolution sentence kernel based on word2vec embeddings. They smooth the delta word kernel to capture the semantic similarity of words. The similarity between sentences is obtained by combining the similarity of all the phrases. Although these approaches go beyond label matching, there is a high computational cost due to the calculation of distance between all possible pairs of words in the sentences.

Bleik et al. [3] used the graph kernel approach to compare biomedical articles represented as graphs. They mapped the biomedical documents into concept graphs using Unified Medical Language System (UMLS) database and used graph kernel functions to compute the similarity between the text documents. Gonçalves and Quaresma [11] represented text documents as graphs using discourse representation theory. The graph-based semantic representations of documents are then compared using a graph kernel based on direct product graph. Nikolentzos et al. [21] used a modified shortest path graph kernel to compute the similarity of two text documents represented as graph-of-words. The graph-of-words representation of text document is converted to shortest path graph. The edges in the shortest path graph connect vertices if the shortest distance between them is not above a threshold d and each edge is labelled by the inverse of the shortest distance between the vertices that the edge connects. The similarity between the text documents is based on the number of matching terms and takes into account the distance between the terms in the documents. Our work differs from theirs in the graph-based representation of the text documents and the information considered while calculating the similarity between graphs. The two main advantages of using the proposed enriched co-occurrence graph representation of text for document similarity are (1) it considers the relevant content of each document as the terms and associations are weighted ensuring that irrelevant information in text is not taken into account while calculating the similarity between documents that affects the categorization of documents and (2) it matches synonymous terms and similar patterns.

3 Graph representation of text

In this section, we introduce the proposed graph representation of text.

3.1 Proposed weighted co-occurrence graph representation

The first step in the proposed text classification approach is the construction of a graph for each of the documents to be classified. We represent each text document as a weighted co-occurrence graph. The nodes represent the unique terms in the document and the edges connect nodes that co-occur within a predefined sliding window of fixed size. We weight the nodes and the edges based on the relevance of the terms and their associations respectively.

The relevance of the terms is determined using the supervised term weight factor—supervised relevance weight (srw) that we proposed in [28]. The supervised term weight factor gives higher weight to terms that help in distinguishing the documents in different classes. It is calculated from the information on the distribution of the training documents in the predefined classes.

The calculation of srw for each term t involves three steps. Step one is the calculation of $class\_rel\_prob(t, C_i)$ for each class C_i which is given in Eq. (1) where a, b and c denote the number of documents in class C_i that contain the term t, the number of documents in class C_i that do not contain the term t and the number of documents not in class C_i that contain the term t respectively. Hence, $class\_rel\_prob(t, C_i)$ determines the concentration of term t in class C_i compared to its concentration in other classes. The number of documents in class C_i that do not contain the term t is considered so that higher weights are not assigned to terms in classes having more training documents.

$$class\_rel\_prob(t, C_i) = \log_2\left(2 + \frac{a}{\max(1, c)}\right) \times \log_2\left(2 + \frac{a}{\max(1, b)}\right) \tag{1}$$

Step two is the calculation of the average densities of the term t in the classes as shown in Eq. (2) where C is the total number of classes and N_i is the total number of documents in class C_i. The sum of densities of the term t in the classes is divided by the number of classes to obtain the average density of the term t. The inverse of the average densities of the term is used in the supervised term weight calculation for reducing the over weighting of commonly occurring terms.

$$avg\_density(t) = \frac{\sum_{i=1}^{C} \left(\frac{a}{N_i} \right)}{C} \tag{2}$$

Finally, step three calculates srw for a term t as shown in Eq. (3). The $class\_rel\_prob(t, C_i)$ value for a term t is determined for each class C_i. $max\_class\_rel(t)$ is the maximum of the $class\_rel\_prob(t, C_i)$ values for a term t.

$$\text{srw}(t) = max\_class\_rel(t) \times \log_{10}\left(\frac{1}{avg\_density(t)} \right) \tag{3}$$

The weights of nodes and edges are calculated using supervised term weights. The weight of each node v (representing term t), denoted by $w_{node}(v)$, is calculated as in Eq. (4) where $f(t)$ is the frequency of term t in the document that the graph represents.

$$w_{node}(v) = f(t) \times \text{srw}(t) \tag{4}$$

We use edge weights to represent the strength of the association between the co-occurring words. For two connecting nodes v_i and v_j representing terms t_i and t_j respectively, with an edge $e = (v_i, v_j) \in E$, the weight for e, denoted by $w_{edge}(e)$, is calculated as

$$w_{edge}(e) = \lambda \times \sqrt{\text{srw}(t_i) \times \text{srw}(t_j)} \tag{5}$$

where λ is the number of times that the terms t_i and t_j co-occur in the document within a fixed size sliding window.

The advantage of using the supervised term weight factor for weighting the nodes and edges is the reduction in the weights of the irrelevant nodes and relationships. Hence, the information in our proposed enriched graph representation enables the similarity measure to take into account the relevant terms and associations shared between documents. After representing each document by a weighted co-occurrence graph, the next step is the enrichment of the co-occurrence graph using a similarity matrix, which is explained in Sect. 4.

4 Automatic enrichment of graphs

There are many methods to calculate the semantic similarity between words, such as ontology, thesaurus and word embedding-based approaches. We utilise the similarities between words obtained using word embeddings to build a word similarity matrix. We used a similarity matrix that only contains similarity values greater than a threshold T (set as 0.9 in our experiments). This similarity matrix is used to automatically enrich the weighted co-occurrence graph built for each document in order to add similar nodes (called as node enrichment) and similar edges/patterns (called as edge enrichment). For example, during node enrichment, if the co-occurrence graph has a node that denotes the term 'likes', then the terms in the similarity matrix that are similar to 'likes' and are not in the co-occurrence graph are added automatically as new nodes; these nodes are then assigned weights based on the weights of similar nodes. During edge enrichment, if the co-occurrence graph has an edge connecting 'likes' and 'hot' corresponding to the pattern 'likes hot', similar patterns such as 'loves warm' are added automatically by connecting the nodes 'loves' and 'warm' and the edges are assigned weights based on the weights of similar patterns. The automatic enrichment of graphs consisting of node enrichment and edge enrichment is explained in Sects. 4.1 and 4.2 respectively. The steps to convert the weighted co-occurrence graph to an enriched graph is illustrated in Fig. 1. The algorithm for automatic enrichment of the weighted co-occurrence graph using a similarity matrix **S** is described below (in Algorithm 1).

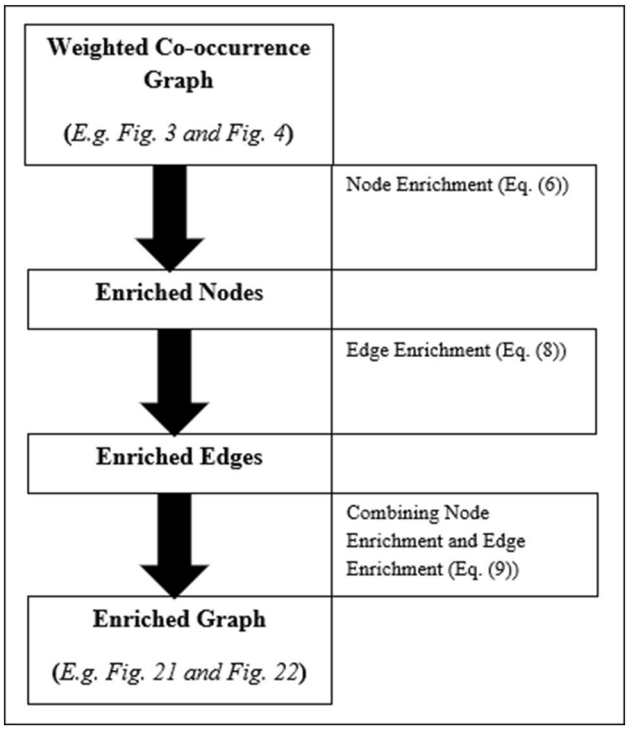

Fig. 1 Steps to convert the weighted co-occurrence graph to an enriched graph

Algorithm 1: Automatic enrichment of weighted co-occurrence graph using similarity matrix $\mathbf{S}$

Input: G - weighted co-occurrence graph
Output: $\widehat{\mathbf{M}}$ - adjacency matrix of the enriched graph
Initialize: $\mathbf{n} = [n_1, \ldots, n_p]$ - node vector where n_i is the weight of the nodes
1: **procedure** ENRICH(G)
2: $\quad \widehat{\mathbf{n}} = \mathbf{n} \times \mathbf{S}$
3: $\quad \mathbf{M_1} = \mathbf{A} \times \mathbf{S}$ ▷ $\mathbf{A}$ is the adjacency matrix of G
4: $\quad \mathbf{M_2} = \mathbf{M_1} \times \mathbf{S}$
5: $\quad \mathbf{M} = \mathbf{M_1} + (\mathbf{B_2} - \mathbf{B_1}) \odot \mathbf{M_2}$ ▷ $\mathbf{B_1}$ and $\mathbf{B_2}$ are the boolean matrices of $\mathbf{M_1}$ and $\mathbf{M_2}$ respectively
6: $\quad \widehat{\mathbf{M}} = \mathbf{M} + \begin{bmatrix} \widehat{n}_1 & 0 & 0 & \ldots & 0 \\ 0 & \widehat{n}_2 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \widehat{n}_p \end{bmatrix}$
7: $\quad$ **return** $\widehat{\mathbf{M}}$
8: **end procedure**

4.1 Node enrichment

The similarity matrix $\mathbf{S} = \left(s_{ij}\right)_{p \times p}$ is a $p \times p$ matrix where p is the number of unique terms in the training documents and s_{ij} is the similarity between the word embeddings of terms t_i and t_j obtained using Word2Vec. The nodes in the document graph are represented by a node vector $\mathbf{n} = [n_1, \ldots, n_p]$ where n_i is the weight of the nodes calculated using Eq. (4). The enriched node vector is obtained by multiplying $\mathbf{n}$ by the similarity matrix as shown in Eq. (6). Hence, node enrichment is done with a semantic kernel [29] that uses supervised term weighting and a semantic matrix built from word embedding-based semantic similarity between words.

$$\widehat{\mathbf{n}} = \mathbf{n} \times \mathbf{S} \tag{6}$$

In Eq. (6), new nodes that are semantically similar to the existing nodes are added if not present in the graph, and the weights of the nodes are assigned/updated as given below in Eq. (7). Let $\Theta(v_i)$ be the set of the nodes which are semantically similar to node v_i that represents the term t_i. The newly added nodes have the initial weight denoted as $w_{node}(v_i)$ equal to 0 and are assigned weights based on the weights of the similar nodes. The weights of the existing nodes are updated if there are similar nodes in the graph.

$$\widehat{w}_{node}(v_i) = w_{node}(v_i) + \sum_{v_j \in \Theta(v_i)} (s_{ij} \times w_{node}(v_j)) \tag{7}$$

4.2 Edge enrichment

The proposed graph enrichment method helps in considering not only the semantic terms shared but also the relationships. The edge enrichment is done so that document similarity goes beyond exact matching of patterns in documents. The edge enrichment method uses similarity matrix to transform the adjacency matrix representation of a document graph to an enriched representation.

The weighted co-occurrence graph is represented as an adjacency matrix $\mathbf{A}$. Edge enrichment is done by utilising the adjacency matrix $\mathbf{A}$ and the similarity matrix $\mathbf{S}$. During edge enrichment, similar edges/patterns are added and the weights of the edges are assigned/updated. Edge enrichment using the similarity matrix $\mathbf{S}$ converts the adjacency matrix $\mathbf{A}$ to $\mathbf{M}$ as given below in Eq. (8) where $\mathbf{M_1} = \mathbf{A} \times \mathbf{S}$ and $\mathbf{M_2} = \mathbf{M_1} \times \mathbf{S}$. It is ensured that $\mathbf{M_1}$ is a symmetric matrix before $\mathbf{M_2}$ is calculated. $\mathbf{M_1}$ and $\mathbf{M_2}$ are converted to boolean matrices $\mathbf{B_1}$ and $\mathbf{B_2}$ respectively by setting non zero values in $\mathbf{M_1}$ and $\mathbf{M_2}$ to 1. We obtain only the newly added elements of $\mathbf{M_2}$, i.e. the zero elements in $\mathbf{M_1}$ that are changed to non-zero elements in $\mathbf{M_2}$, by computing $\left(\mathbf{B_2} - \mathbf{B_1}\right) \odot \mathbf{M_2}$ where $\odot$ corresponds to the element-wise product of matrices.

$$\mathbf{M} = \mathbf{M_1} + \left(\mathbf{B_2} - \mathbf{B_1}\right) \odot \mathbf{M_2} \tag{8}$$

Equation (9) shows the computation of the final adjacency matrix $\widehat{\mathbf{M}}$ which is obtained by adding $\mathbf{M}$ with diagonal matrix from the enriched node vector $\widehat{\mathbf{n}} = [\widehat{n_1}, \ldots, \widehat{n_p}]$. $\widehat{\mathbf{M}}$ is the adjacency matrix representation of the enriched graph.

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	1	0.9	0.0	0.0	0.0	0.0	0.0
drinks	0.9	1	0.0	0.0	0.0	0.0	0.0
hot	0.0	0.0	1	0.0	0.0	0.0	0.8
john	0.0	0.0	0.0	1	0.0	0.0	0.0
likes	0.0	0.0	0.0	0.0	1	0.9	0.0
loves	0.0	0.0	0.0	0.0	0.9	1	0.0
warm	0.0	0.0	0.8	0.0	0.0	0.0	1

Fig. 2 The similarity matrix for our toy example using seven words where the values correspond to the similarity between words

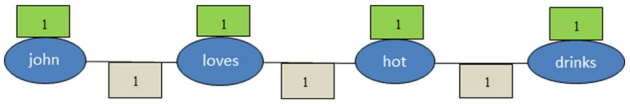

Fig. 3 Co-occurrence graph representation of 'John loves hot drinks' with the initial weights of nodes and edges assumed as 1

Fig. 4 Co-occurrence graph representation of 'John likes warm beverages' with the initial weights of nodes and edges assumed as 1

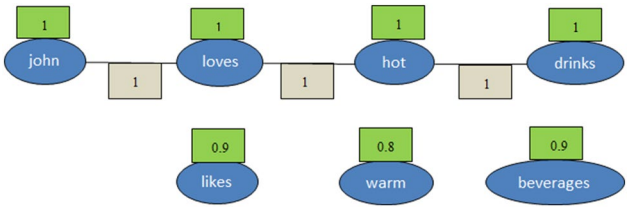

Fig. 5 Node enrichment of graph of 'John loves hot drinks' that adds similar nodes and this step corresponds to Eq. (6)

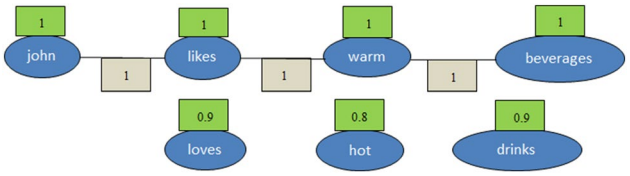

Fig. 6 Node enrichment of graph of 'John likes warm beverages' that adds similar nodes and this step corresponds to Eq. (6)

$$\widehat{\mathbf{M}} = \mathbf{M} + \begin{bmatrix} \widehat{n}_1 & 0 & 0 & \dots & 0 \\ 0 & \widehat{n}_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \widehat{n}_p \end{bmatrix} \tag{9}$$

4.3 Example to illustrate node enrichment and edge enrichment

Initially, a similarity matrix is built from the unique words in the training documents. Suppose the unique words obtained from the training documents are 'beverages', 'drinks', 'hot', 'john', 'likes', 'loves' and 'warm'. The similarity matrix for the toy example is given in Fig. 2. The two documents to be compared in the example are 'John loves hot drinks' and 'John likes warm beverages'. Even though the sentences

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	0.0	0.0	0.0	0.0	0.0
drinks	0.0	0.0	1.0	0.0	0.0	0.0	0.0
hot	0.0	1.0	0.0	0.0	0.0	1.0	0.0
john	0.0	0.0	0.0	0.0	0.0	1.0	0.0
likes	0.0	0.0	0.0	0.0	0.0	0.0	0.0
loves	0.0	0.0	1.0	1.0	0.0	0.0	0.0
warm	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 7 Adjacency matrix representation of 'John loves hot drinks'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	0.0	0.0	0.0	0.0	1.0
drinks	0.0	0.0	0.0	0.0	0.0	0.0	0.0
hot	0.0	0.0	0.0	0.0	0.0	0.0	0.0
john	0.0	0.0	0.0	0.0	1.0	0.0	0.0
likes	0.0	0.0	0.0	1.0	0.0	0.0	1.0
loves	0.0	0.0	0.0	0.0	0.0	0.0	0.0
warm	1.0	0.0	0.0	0.0	1.0	0.0	0.0

Fig. 8 Adjacency matrix representation of 'John likes warm beverages'

are similar, similarity measures based on term overlap/keyword matching do not give accurate similarity value as only one word in the documents match. The proposed automatic graph enrichment method builds enriched graphs with similar structures for documents with similar meaning resulting in accurate similarity calculation.

The initial weighted co-occurrence graph representations of the documents (obtained with a predefined sliding window of size 2) are given below in Figs. 3 and 4. The initial weights of the nodes and edges are assumed as 1 in this example. In actual cases, the weights of nodes and edges are calculated as in Eqs. (4) and (5) respectively.

During node enrichment (that corresponds to Eq. (6)), new nodes are added with weights based on the weights of the similar nodes as shown in Figs. 5 and 6. In Fig. 5, the nodes corresponding to the words 'likes', 'warm' and 'beverages' are the newly added nodes which are semantically similar to the existing nodes that represent the words 'loves', 'hot' and 'drinks' respectively. Similarly, in Fig. 6, the nodes that denote the words such as 'loves', 'hot' and 'drinks' are the nodes added during the node enrichment step. These newly added nodes are assigned weights based on the weights of the similar nodes in the graph. For example, in Fig. 5, the node 'likes' is assigned a weight of 0.9 which is obtained by computing the product of the weight of the similar node 'loves' and the value of its similarity to the node.

The graph is represented as an adjacency matrix before enriching the edges. The adjacency matrices in Figs. 7 and 8 denote the graphs in Figs. 5 and 6 respectively. These are symmetric matrices with values that correspond to the weights of edges in the graphs.

Fig. 9 Matrix $\mathbf{M_1}$ for 'John loves hot drinks'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	0.9	0.0	0.0	0.0	0.0
drinks	0.0	0.0	1.0	0.0	0.0	0.0	0.8
hot	0.9	1.0	0.0	0.0	0.9	1.0	0.0
john	0.0	0.0	0.0	0.0	0.9	1.0	0.0
likes	0.0	0.0	0.9	0.9	0.0	0.0	0.0
loves	0.0	0.0	1.0	1.0	0.0	0.0	0.8
warm	0.0	0.8	0.0	0.0	0.0	0.8	0.0

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	0.8	0.0	0.0	0.0	1.0
drinks	0.0	0.0	0.0	0.0	0.0	0.0	0.9
hot	0.8	0.0	0.0	0.0	0.8	0.0	0.0
john	0.0	0.0	0.0	0.0	1.0	0.9	0.0
likes	0.0	0.0	0.8	1.0	0.0	0.0	1.0
loves	0.0	0.0	0.0	0.9	0.0	0.0	0.9
warm	1.0	0.9	0.0	0.0	1.0	0.9	0.0

Fig. 10 Matrix $\mathbf{M_1}$ for 'John likes warm beverages'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	0.9	0.0	0.0	0.0	0.72
drinks	0.0	0.0	1.64	0.0	0.0	0.0	1.60
hot	1.8	1.81	0.0	0.0	1.8	1.81	0.0
john	0.0	0.0	0.0	0.0	1.8	1.81	0.0
likes	0.0	0.0	0.9	0.9	0.0	0.0	0.72
loves	0.0	0.0	1.64	1.0	0.0	0.0	1.60
warm	0.72	0.8	0.0	0.0	0.72	0.8	0.0

Fig. 11 Matrix $\mathbf{M_2}$ for 'John loves hot drinks'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	1.6	0.0	0.0	0.0	1.64
drinks	0.0	0.0	0.72	0.0	0.0	0.0	0.9
hot	0.8	0.72	0.0	0.0	0.8	0.72	0.0
john	0.0	0.0	0.0	0.0	1.81	1.8	0.0
likes	0.0	0.0	1.6	1.0	0.0	0.0	1.64
loves	0.0	0.0	0.72	0.9	0.0	0.0	0.9
warm	1.81	1.8	0.0	0.0	1.81	1.8	0.0

Fig. 12 Matrix $\mathbf{M_2}$ for 'John likes warm beverages'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	1.0	0.0	0.0	0.0	0.0
drinks	0.0	0.0	1.0	0.0	0.0	0.0	1.0
hot	1.0	1.0	0.0	0.0	1.0	1.0	0.0
john	0.0	0.0	0.0	0.0	1.0	1.0	0.0
likes	0.0	0.0	1.0	1.0	0.0	0.0	0.0
loves	0.0	0.0	1.0	1.0	0.0	0.0	1.0
warm	0.0	1.0	0.0	0.0	0.0	1.0	0.0

Fig. 13 Matrix $\mathbf{B_1}$ for 'John loves hot drinks'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	1.0	0.0	0.0	0.0	1.0
drinks	0.0	0.0	0.0	0.0	0.0	0.0	1.0
hot	1.0	0.0	0.0	0.0	1.0	0.0	0.0
john	0.0	0.0	0.0	0.0	1.0	1.0	0.0
likes	0.0	0.0	1.0	1.0	0.0	0.0	1.0
loves	0.0	0.0	0.0	1.0	0.0	0.0	1.0
warm	1.0	1.0	0.0	0.0	1.0	1.0	0.0

Fig. 14 Matrix $\mathbf{B_1}$ for 'John likes warm beverages'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	1.0	0.0	0.0	0.0	1.0
drinks	0.0	0.0	1.0	0.0	0.0	0.0	1.0
hot	1.0	1.0	0.0	0.0	1.0	1.0	0.0
john	0.0	0.0	0.0	0.0	1.0	1.0	0.0
likes	0.0	0.0	1.0	1.0	0.0	0.0	1.0
loves	0.0	0.0	1.0	1.0	0.0	0.0	1.0
warm	1.0	1.0	0.0	0.0	1.0	1.0	0.0

Fig. 15 Matrix $\mathbf{B_2}$ for 'John loves hot drinks'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	1.0	0.0	0.0	0.0	1.0
drinks	0.0	0.0	1.0	0.0	0.0	0.0	1.0
hot	1.0	1.0	0.0	0.0	1.0	1.0	0.0
john	0.0	0.0	0.0	0.0	1.0	1.0	0.0
likes	0.0	0.0	1.0	1.0	0.0	0.0	1.0
loves	0.0	0.0	1.0	1.0	0.0	0.0	1.0
warm	1.0	1.0	0.0	0.0	1.0	1.0	0.0

Fig. 16 Matrix $\mathbf{B_2}$ for 'John likes warm beverages'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	0.9	0.0	0.0	0.0	0.72
drinks	0.0	0.0	1.0	0.0	0.0	0.0	0.8
hot	0.9	1.0	0.0	0.0	0.9	1.0	0.0
john	0.0	0.0	0.0	0.0	0.9	1.0	0.0
likes	0.0	0.0	0.9	0.9	0.0	0.0	0.72
loves	0.0	0.0	1.0	1.0	0.0	0.0	0.8
warm	0.72	0.8	0.0	0.0	0.72	0.8	0.0

Fig. 17 Matrix $\mathbf{M}$ for 'John loves hot drinks'

Edge enrichment carried out utilising the similarity matrix converts the adjacency matrix to matrix M as shown in Figs. 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18. During edge enrichment, similar patterns are added and weighted based on the weights of the existing edges. For example, for the 'hot drinks' edge in the graph in Fig. 5, the patterns that are similar to it such as 'hot beverages' and 'warm drinks' are added with the initial transformation using similarity

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.0	0.0	0.8	0.0	0.0	0.0	1.0
drinks	0.0	0.0	0.72	0.0	0.0	0.0	0.9
hot	0.8	0.72	0.0	0.0	0.8	0.72	0.0
john	0.0	0.0	0.0	0.0	1.0	0.9	0.0
likes	0.0	0.0	0.8	1.0	0.0	0.0	1.0
loves	0.0	0.0	0.72	0.9	0.0	0.0	0.9
warm	1.0	0.9	0.0	0.0	1.0	0.9	0.0

Fig. 18 Matrix **M** for 'John likes warm beverages'

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	0.9	0.0	0.9	0.0	0.0	0.0	0.72
drinks	0.0	1.0	1.0	0.0	0.0	0.0	0.8
hot	0.9	1.0	1.0	0.0	0.9	1.0	0.0
john	0.0	0.0	0.0	1.0	0.9	1.0	0.0
likes	0.0	0.0	0.9	0.9	0.9	0.0	0.72
loves	0.0	0.0	1.0	1.0	0.0	1.0	0.8
warm	0.72	0.8	0.0	0.0	0.72	0.8	0.8

Fig. 19 Final adjacency matrix representation of 'John loves hot drinks' obtained after graph enrichment, corresponding to $\widehat{\mathbf{M}}$ in Eq. (9)

	beverages	*drinks*	*hot*	*john*	*likes*	*loves*	*warm*
beverages	1.0	0.0	0.8	0.0	0.0	0.0	1.0
drinks	0.0	0.9	0.72	0.0	0.0	0.0	0.9
hot	0.8	0.72	0.8	0.0	0.8	0.72	0.0
john	0.0	0.0	0.0	1.0	1.0	0.9	0.0
likes	0.0	0.0	0.8	1.0	1.0	0.0	1.0
loves	0.0	0.0	0.72	0.9	0.0	0.9	0.9
warm	1.0	0.9	0.0	0.0	1.0	0.9	1.0

Fig. 20 Final adjacency matrix representation of 'John likes warm beverages' obtained after graph enrichment, corresponding to $\widehat{\mathbf{M}}$ in Eq. (9)

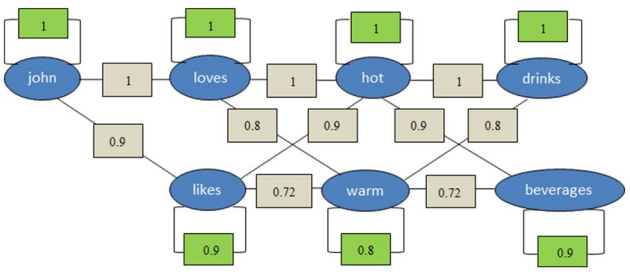

Fig. 21 Enriched co-occurrence graph of 'John loves hot drinks' obtained after graph enrichment

matrix (which correspond to elements in $\mathbf{M_1}$ in Eq. (8)) as shown in Fig. 9. In the subsequent transformation using similarity matrix, the 'warm beverages' edge is added (which corresponds to an element in **M** in Eq. (8)) as given

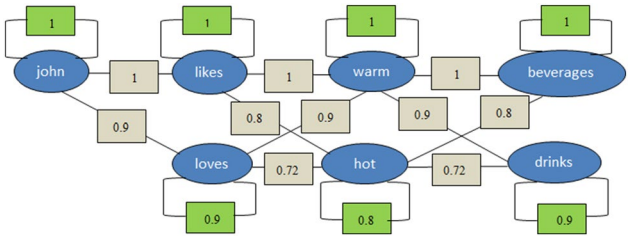

Fig. 22 Enriched co-occurrence graph of 'John likes warm beverages' obtained after graph enrichment

in Fig. 17. Similarly, for the 'warm beverages' edge in the graph in Fig. 6, the patterns that are similar to it such as 'warm drinks' and 'hot beverages' are added with the initial transformation using similarity matrix (which correspond to elements in $\mathbf{M_1}$ in Eq. (8)) as shown in Fig. 10. In the subsequent transformation using similarity matrix, the 'hot drinks' edge is added (which corresponds to an element in **M** in Eq. (8)) as given in Fig. 18. The boolean matrices in Figs. 13, 14, 15 and 16 are obtained by setting the non zero values of matrices in Figs. 9, 10, 11 and 12 to one. The final adjacency matrix representations of the graphs (which corresponds to $\widehat{\mathbf{M}}$ in Eq. (9)) are shown in Figs. 19 and 20. The adjacency matrix representations are converted to the enriched graphs as given in Figs. 21 and 22. Hence, the two documents have similar enriched graph structures which lead to accurate calculation of similarity between the text documents. The advantage of using graph kernels for text similarity is that we can compare terms (represented by nodes) and patterns (represented by edges) in documents effectively and efficiently. The proposed graph enrichment enables the graph kernels to go beyond exact matching of terms and patterns.

5 Graph kernel-based text classification

In this section, we explain the calculation of similarity between the enriched graph representations of text and then briefly describe the classification pipeline.

5.1 Graph kernels for measuring document similarity

The kernel approach allows the extension of linear algorithms to non-linear models, and helps in the application of algorithms to structured representation such as strings, trees and graphs. The kernel function is a dot product in an implicit feature space. This helps in replacing the dot products in kernel machines and hence, kernel methods solve the problem of direct application of existing pattern recognition algorithms to graphs [7].

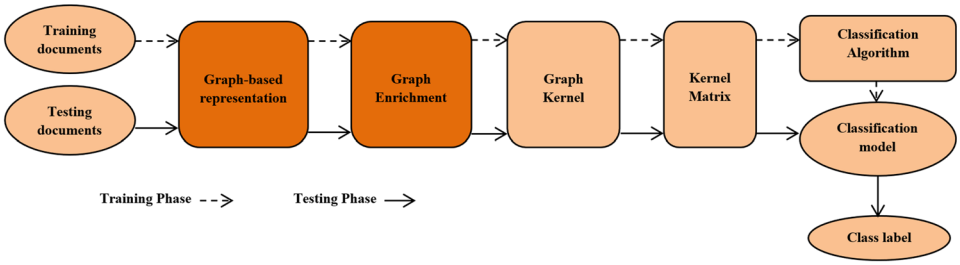

Fig. 23 Graph kernel-based text classification pipeline

A kernel measures the similarity between objects. The kernel matrix created should satisfy the two important mathematical properties of matrix symmetry and positive semi-definiteness [33]. To compare two documents d_i and d_j represented by enriched weighted co-occurrence graphs $G_i = (V_i, E_i)$ and $G_j = (V_j, E_j)$ respectively, we use an edge walk kernel [6, 21] as shown in Eq. (10) to compare the edges in both the graphs. The edge walk kernel is explained below in Eqs. (10), (11) and (12). The normalization factor is the product of the frobenius norms of the adjacency matrices A_i and A_j of the graphs G_i and G_j respectively so that the similarity value is not affected by the number of nodes and edges in the graph. $||A_i||_F$ and $||A_j||_F$ correspond to the frobenius norms of the adjacency matrices of the graphs G_i and G_j respectively. Let u_i and v_i be the vertices that belong to the set of vertices V_i in G_i, e_i be the edge linking u_i and v_i in G_i, u_j and v_j be the vertices that belong to the set of vertices V_j in G_j, e_j be the edge connecting u_j and v_j in G_j. $k^{(1)}_{walk}$ is a kernel that compares edge walks of length 1 in the graphs G_i and G_j. It is the product of the kernel function on the edge and the two nodes that the edge connects as defined in Eq. (11).

$$k(d_i, d_j) = \frac{\sum_{e_i \epsilon E_i, e_j \epsilon E_j} k^{(1)}_{walk}(e_i, e_j)}{||A_i||_F \times ||A_j||_F} \tag{10}$$

$$k^{(1)}_{walk}(e_i, e_j) = k_{node}(u_i, u_j) \times k_{edge}(e_i, e_j) \times k_{node}(v_i, v_j) \tag{11}$$

A delta kernel function, k_{node}, is used for comparing the vertices and is equal to 1 if the terms corresponding to the vertices are the same and 0 if the terms are different. k_{edge} is a kernel function for comparing the edges in the graphs and is defined in Eq. (12). It is the product of the weight of the edge e_i in G_i denoted as $w_{edge}(e_i)$ and the weight of the edge e_j in G_j denoted as $w_{edge}(e_j)$. Hence, the numerator in Eq. (10) is equivalent to the sum of the elements in the element-wise product of the adjacency matrices A_i and A_j.

$$k_{edge}(e_i, e_j) = \begin{cases} w_{edge}(e_i) \times w_{edge}(e_j) & \text{if } e_i \epsilon E_i \wedge e_j \epsilon E_j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

The delta kernels are positive definite. The kernel $k^{(1)}_{walk}$ is a product of the delta kernels multiplied by a positive number, thus preserving positive definiteness. The edge walk kernel function is a sum of the positive definite kernels divided by a positive number. Hence, the positive definiteness is preserved and it is a valid kernel. The similarity between every pair of graphs is determined using the edge walk kernel, and the values obtained are used to build a kernel matrix. The most common kernel-based classifier is SVM [16]. The kernel matrix is then used with SVM classifier to learn and predict the classes of the document. The worst case time complexity of the graph kernel is $O(n + m)$ where n is the number of unique nodes (or the size of the vocabulary) and m is the number of edges. Hence, it is higher than that of the document similarity measure with bag-of-words representation (unigram features) whose time complexity is $O(n)$.

5.2 Graph kernel-based text classification pipeline

The proposed graph kernel-based text classification pipeline is shown in Fig. 23. The documents are initially represented as weighted co-occurrence graphs where the nodes represent the unique terms and the edges represent the association between the words co-occurring within a predefined sliding window of size 2. The supervised term weight factor is utilised to assign weight to nodes and edges. These graphs are enriched automatically using a similarity matrix built with similarity values obtained using word embeddings. A graph kernel based on edge matching is employed to calculate the similarity between a pair of documents. The similarity values are then used to build a kernel matrix. The kernel matrix is fed to a SVM to learn and predict the classes of the documents.

6 Experiments and results

In this section, we describe the experiments performed on sentiment analysis and topic classification tasks to evaluate the performance of the proposed knowledge-driven graph similarity measure for text classification. The datasets used are briefly explained below.

– *Sentence polarity dataset* This dataset consists of 5331 positive and 5331 negative movie reviews [23].

- *Subjectivity dataset* This dataset consists of 5000 subjective and 5000 objective sentences on movie reviews labelled according to their subjectivity status [22].
- *News* This dataset is a collection of 32,602 short text documents which are news collected from RSS feeds of the websites—nyt.com, usatoday.com and reuters.com and classified based on their topics. The topics are sports, business, US, health, sci&tech, world and entertainment. The document consists of the title, description, link, id, data, source and category of the news. We have used only the description and category of the news [34].
- *Multi-domain sentiment dataset* This dataset consists of 8000 product reviews obtained from amazon.com where the products are books, dvd, electronics and kitchen [4]. There are 1000 positive reviews and 1000 negative reviews for each of the four product domains.
- *20 Newsgroups*[1] The 20 Newsgroups dataset contains 20,000 newsgroup documents classified into 20 different categories.

In the proposed method, each document is represented as a weighted co-occurrence graph where the nodes represent the unique terms in the document and edges link words that co-occur within a predefined sliding window. The weight of the node is stored in the self-loop which corresponds to the importance of the term based on its relevance in classifying the text documents. Nodes that correspond to unimportant terms have lower weight than nodes that represent the main content of the document. Similarly, the edges that connect co-occurring words have weights that are dependent on the relevance of the co-occurring words. The graphs are enriched using a word similarity matrix that contains similarities greater than or equal to 0.9. The text8 corpus (obtained from Wikipedia)[2] is used to build the word2vec model for deriving the word embedding vectors. The similarity values of the top five similar words for each unique word in the training set are used to create the similarity matrix. The threshold for the similarity between the word vectors is set as 0.9 to obtain the closely related terms or synonyms. The graph enrichment process can become slow with increase in the size and density of the similarity matrix. The different ways to increase the speed of the enrichment process are given below:

(i) Set a threshold for the similarity values in the similarity matrix. This would result in a sparse matrix reducing the time for matrix operations in graph enrichment.

(ii) Build a similarity matrix with only the most relevant features which would reduce the size of the matrix.

(iii) Also, the graph enrichment process for different documents could be done in parallel since they are not dependent on each other. This would make it considerably faster.

We have compared the proposed approach with linear kernel, cosine similarity, Sorensen similarity [24], Tanimoto similarity [24], radial basis function (RBF) kernel, class meaning kernel (CMK) [2], class weighting kernel (CWK) [1] and the shortest path graph kernel (spgk) [6, 21] method which is a graph kernel approach for text classification with a different graph representation of text. The linear kernel, cosine similarity and RBF kernel are computed with tf-idf weighted feature vectors of documents. The Sorensen similarity and Tanimoto similarity measures are calculated with boolean vectors of documents. In CMK and CWK, the documents are represented as tf weighted vectors and the semantic smoothing is then done using semantic matrix built from meaning values of terms and supervised term weights respectively. In the shortest path graph kernel method, the co-occurrence graph is converted to a shortest path graph with edges connecting nodes that have the shortest distance not above a threshold d and the edges are labelled by the inverse of the shortest distance between the nodes. The evaluation metrics used to assess the performance of text classification are precision, recall and F1 score. The performance of shortest path graph kernel with different values of d has been evaluated. The proposed method is also experimented with co-occurrence graphs built using predefined sliding window w of sizes 2, 3 and 4. The kernel matrices are built with the similarity values obtained using linear kernel, cosine similarity, Sorensen similarity, Tanimoto similarity, RBF kernel, CMK, CWK, spgk and proposed method. The row in the kernel matrix represents the similarity of a document to be classified with the documents in the training set. Each kernel matrix is fed to SVM to evaluate the performance of text classification using the similarity measure.

The proposed similarity measure is implemented using python. The networkx, gensim and scikit-learn are the python packages used to create the graphs, word2vec model and the kernel SVM respectively. Table 1 shows the precision, recall and F1 scores obtained for the sentiment classification datasets. Table 2 shows the precision, recall and F1 scores obtained for the topic classification tasks. Table 3 shows the performance of the proposed method with co-occurrence graphs built using predefined sliding window w of sizes 2, 3 and 4. The results reported in these tables are obtained by tenfold cross validation except for the 20 Newsgroups dataset that has a standard train/test split. The validation set is 20 percent of the training set and is used to optimize the value of the parameter C in SVM. The best value of C from the set of values {0.01,0.1,1,10,100,1000} is then used to classify the documents in the testing set.

Tables 4 and 5 compare the classification performances (using train/test split) of the proposed method and the

[1] http://ana.cachopo.org/datasets-for-single-label-text-categorization

[2] http://mattmahoney.net/dc/textdata.html

Table 1 Precision, recall and F1 scores for sentiment classification tasks using different similarity measures

Dataset	Metric	Linear	Cosine	Sorensen	Tanimoto	RBF	Spgk				Proposed method
							d=1	d=2	d=3	d=4	
Polarity	Precision	77.15	77.15	76.65	77.36	77.09	77.13	77.18	77.43	77.77	**81.47**
	Recall	77.12	77.11	76.60	77.33	77.07	77.10	77.14	77.39	77.74	**81.42**
	F1	77.12	77.11	76.59	77.32	77.06	77.10	77.13	77.39	77.74	**81.42**
Subjectivity	Precision	90.98	91.12	90.21	90.86	91.07	90.82	91.02	90.85	90.85	**92.74**
	Recall	90.94	91.08	90.18	90.84	91.03	90.80	91.00	90.82	90.83	**92.73**
	F1	90.94	91.08	90.18	90.84	91.03	90.80	91.00	90.82	90.83	**92.73**
Books	Precision	80.58	80.91	79.88	79.85	80.29	80.85	81.13	81.11	80.55	**86.12**
	Recall	80.44	80.77	79.69	79.74	80.09	80.74	81.09	81.04	80.49	**86.04**
	F1	80.42	80.79	79.66	79.72	80.06	80.72	81.09	81.03	80.48	**86.04**
Dvd	Precision	81.70	82.61	79.67	80.59	81.88	80.63	81.86	81.30	81.34	**87.40**
	Recall	81.55	82.50	79.50	80.50	81.70	80.50	81.75	81.25	81.25	**87.20**
	F1	81.53	82.49	79.47	80.49	81.68	80.48	81.73	81.24	81.24	**87.19**
Electronics	Precision	80.72	80.25	81.07	82.36	80.29	83.07	83.38	84.13	84.05	**86.01**
	Recall	80.45	80.05	81.00	82.30	79.95	83.00	83.30	84.05	84.00	**85.90**
	F1	80.41	80.01	80.99	82.29	79.98	82.99	83.29	84.04	83.99	**85.89**
Kitchen	Precision	84.96	85.78	85.18	85.52	85.27	85.78	85.86	85.82	86.07	**90.20**
	Recall	84.90	85.70	84.95	85.35	85.20	85.70	85.75	85.70	85.95	**90.10**
	F1	84.89	85.69	84.92	85.33	85.19	85.69	85.69	85.74	85.94	**90.09**

In the underlined datasets, the improvements of the proposed method over linear kernel are statistically significant at $p < 0.01$ using sign test

Table 2 Precision, recall and F1 scores for topic classification tasks using different similarity measures

Dataset	Metric	Linear	Cosine	Sorensen	Tanimoto	RBF	Spgk				Proposed method
							d=1	d=2	d=3	d=4	
20NG	Precision	80.33	83.44	83.77	83.58	80.52	81.72	81.64	81.41	81.44	**85.10**
	Recall	79.23	83.03	83.27	82.97	78.59	80.92	80.72	80.55	80.56	**84.19**
	F1	79.31	83.03	83.27	82.95	78.94	81.01	80.79	80.59	80.60	**84.36**
News	Precision	82.49	82.89	81.34	81.39	82.63	80.88	80.92	80.91	81.01	**84.39**
	Recall	82.44	82.83	81.29	81.40	82.55	80.85	80.89	80.90	81.00	**84.30**
	F1	82.34	82.76	81.16	81.30	82.40	80.72	80.74	80.74	80.85	**84.20**

In the underlined datasets, the improvements of the proposed method over linear kernel are statistically significant at $p < 0.01$ using sign test

Table 3 Precision, recall and F1 scores for the proposed method with graph representation built using predefined sliding window of different sizes

Dataset	Proposed method								
	w=2			w=3			w=4		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Polarity	**81.47**	**81.43**	**81.42**	81.38	81.35	81.34	81.15	81.11	81.11
Subjectivity	92.74	92.73	92.73	**92.81**	**92.80**	**92.80**	92.63	92.62	92.62
Books	86.12	86.04	86.04	**86.24**	**86.14**	**86.13**	85.75	85.64	85.63
Dvd	87.40	87.20	87.19	**87.54**	**87.35**	**87.34**	87.05	86.90	86.89
Electronics	86.01	85.90	85.89	**86.82**	**86.70**	**86.69**	86.63	86.50	86.49
Kitchen	**90.20**	**90.10**	**90.09**	89.90	89.80	89.79	90.02	89.90	89.89
20NG	85.10	**84.19**	**84.36**	**85.17**	83.91	84.18	85.15	83.54	83.88
News	**84.39**	**84.30**	**84.20**	83.87	83.76	83.65	83.72	83.58	83.44

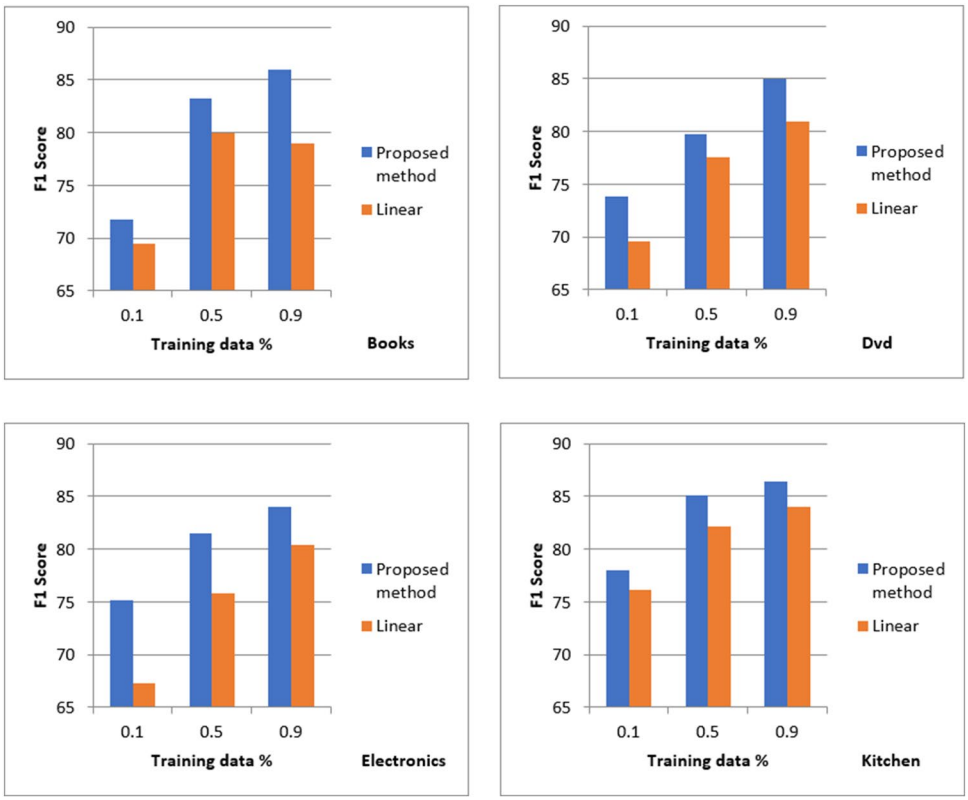

Fig. 24 Classification performance with different sizes of training set

Table 4 Comparison of precision, recall and F1 scores of sentiment classification tasks using supervised semantic kernels

Dataset	Metric	CWK	CWK_{wfs}	CMK	CMK_{wfs}	Proposed method
Polarity	Precision	62.56	77.49	63.16	75.89	**78.38**
	Recall	62.07	77.46	62.68	75.82	**78.36**
	F1	61.69	77.46	62.33	75.81	**78.35**
Subjectivity	Precision	82.83	91.43	81.73	90.10	**91.46**
	Recall	82.75	91.40	81.60	90.10	**91.45**
	F1	82.74	91.40	81.58	90.10	**91.45**
Books	Precision	66.29	72.61	72.21	76.73	**81.96**
	Recall	66.17	72.43	71.93	76.69	**81.95**
	F1	66.11	72.37	71.85	76.69	**81.95**
Dvd	Precision	69.57	78.00	71.93	76.50	**84.32**
	Recall	69.50	78.00	71.50	76.50	**84.25**
	F1	69.47	78.00	71.36	76.50	**84.24**
Electronics	Precision	74.00	79.50	74.02	81.05	**81.51**
	Recall	74.00	79.50	74.00	81.00	**81.50**
	F1	74.00	79.50	73.99	81.00	**81.50**
Kitchen	Precision	75.71	83.27	82.32	83.35	**91.25**
	Recall	75.50	83.25	82.25	83.25	**91.25**
	F1	75.45	83.25	82.24	83.24	**91.25**

supervised semantic kernels i.e. CMK and CWK for sentiment and topic classification tasks. Since CMK and CWK require long training time, the performance is evaluated by splitting the dataset into training and testing set in the 80:20 ratio. The default value of 1 for parameter *C* in SVM is used to classify the documents. In text classification with CMK and CWK, attribute selection (as reported in their experiments [1, 2]) is applied using mutual information to select the best 2000 terms. CWK_{wfs} and CMK_{wfs} correspond to the supervised semantic kernels CWK and CMK without

Table 5 Comparison of precision, recall and F1 scores of topic classification tasks using supervised semantic kernels

Dataset	Metric	CWK	CWK_{wfs}	CMK	CMK_{wfs}	Proposed method
20NG	Precision	73.84	79.76	69.29	73.78	**85.01**
	Recall	73.56	79.19	68.98	73.52	**83.94**
	F1	73.55	79.12	69.00	73.49	**84.15**
News	Precision	71.34	83.04	69.24	78.60	**83.95**
	Recall	71.05	83.15	68.66	78.56	**84.02**
	F1	71.04	83.06	68.27	78.42	**83.89**

Table 6 Advantages of the proposed method. Information considered for the computation of similarity between documents

Information considered	Linear	Cosine	Sorensen	Tanimoto	RBF	spgk	CWK	CMK	Proposed method
Importance of terms based on class information (Supervised term weight)	No	No	No	No	No	No	Yes	Yes	Yes
Co-occurrence information	No	No	No	No	No	Yes	No	No	Yes
Importance of associations	No	No	No	No	No	Yes[a]	No	No	Yes[b]
Incorporation of external knowledge	No	No	No	No	No	No	No	No	Yes
Semantic similarity of terms and associations	No	No	No	No	No	No	No	No	Yes

[a]Based on the distance between words.

[b]Based on the number of co-occurrences and relevance of the terms. Distance is not taken into account since terms that co-occur closely are only considered

performing this feature selection. There is a considerable improvement in the performance of these semantic kernels without feature selection.

The proposed approach significantly outperforms the baseline similarity measures for text classification on all datasets in terms of precision, recall and F1 score as shown in Tables 1, 2, 4 and 5. The highest precision, recall and F1 score for each dataset are highlighted in bold in the tables. Table 6 shows the information considered by the proposed approach for the computation of similarity between documents. The advantage of the proposed graph kernel approach for text classification is that it considers the contextual information and is not based on word independence approach as in vector space models. The similarity measure compares the relevant structural information in the documents and computes the semantic similarity between the documents. This is possible due to the semantic information available in the enriched graph representations of the documents. Table 3 shows that there is no considerable difference in the performance with an increase in the size of the predefined sliding window used to build the co-occurrence graphs. Figure 24 presents the results of document classification (in terms of F1 score) using the proposed method and linear kernel with different proportions of training set such as 0.1, 0.5 and 0.9. It shows that the proposed method consistently outperforms the linear kernel even with a small training set (of 10%).

7 Conclusion

Graph-based representations of text are effective for text classification as they can model the structural information in text, which is required to understand its meaning. Considering the structural information in text when calculating the similarity between documents can improve the performance of text classification. In this paper, we focused on building a text graph model that represents the structural information in text effectively, which helps to compare documents based on their main similar content. Supervised term weighting is utilised to weight the terms and their associations, so that the matching terms and patterns contribute to document similarity based on their relevance. The graph enrichment is carried out with the word similarity matrix to consider semantically similar terms and associations, going beyond exact matching of document content. We employed a graph kernel function that utilises the rich information in the enriched weighted graphs to compute the similarity between text documents accurately for improving the performance of classification task. Our experimental results on sentiment analysis and topic classification tasks show that the proposed graph kernel-based approach for text classification detects and exploits the structural patterns in text to compute the semantic similarity between text documents, resulting in a significant improvement in text classification performance. The similarity matrix used in the enrichment

could be improved by designing it based on the application, in order to utilise domain knowledge and increase the accuracy of the similarity measure. An interesting future work is to use an ontology-based similarity matrix with more accurate similarity values to enrich the graph and consider the similar concepts and relationships in measuring document similarity. The proposed text classification framework can be adapted for different domains by designing the similarity matrix based on the domain. The graph enrichment method can be extended to calculate similarity between text documents for other applications such as document clustering, information retrieval and relevance-based document ranking.

Acknowledgements The authors would like to acknowledge the support from Ulster University through the Vice Chancellor's Research Scholarship (VCRS) Award.

References

1. Altınel B, Diri B, Ganiz MC (2015) A novel semantic smoothing kernel for text classification with class-based weighting. Knowl Based Syst 89:265–277
2. Altınel B, Ganiz MC, Diri B (2015) A corpus-based semantic kernel for text classification by using meaning values of terms. Eng Appl Artif Intell 43:54–66
3. Bleik S, Mishra M, Huan J, Song M (2013) Text categorization of biomedical data sets using graph kernels and a controlled vocabulary. IEEE/ACM Trans Comput Biol Bioinform (TCBB) 10(5):1211–1217
4. Blitzer J, Dredze M, Pereira F (2007) Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Association for Computational Linguistics, Prague, Czech Republic, pp 440–447
5. Bloehdorn S, Basili R, Cammisa M, Moschitti A (2006) Semantic kernels for text classification based on topological measures of feature similarity. In: Sixth International Conference on Data Mining (ICDM'06), pp 808–812
6. Borgwardt KM, Kriegel HP (2005) Shortest-path kernels on graphs. In: Fifth IEEE International Conference on Data Mining (ICDM'05), p 8
7. Bunke H, Riesen K (2011) Recent advances in graph-based pattern recognition with applications in document analysis. Pattern Recognit 44(5):1057–1067
8. Cancedda N, Gaussier E, Goutte C, Renders JM (2003) Word-sequence kernels. J Mach Learn Res 3(Feb):1059–1082
9. Cristianini N, Shawe-Taylor J, Lodhi H (2002) Latent semantic kernels. J Intell Inf Syst 18(2–3):127–152
10. Gärtner T, Flach P, Wrobel S (2003) On graph kernels: Hardness results and efficient alternatives. In: Schölkopf B, Warmuth MK (eds) Learning Theory and Kernel Machines, Springer Berlin, Heidelberg, pp 129–143
11. Gonçalves T, Quaresma P (2009) Using graph-kernels to represent semantic information in text classification. In: Perner P (ed) Machine Learning and Data Mining in Pattern Recognition, Springer Berlin, Heidelberg, pp 632–646
12. Hassan S, Mihalcea R, Banea C (2007) Random walk term weighting for improved text classification. Int J Semant Comput 1(04):421–439
13. Haussler D (1999) Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California, Tech. rep
14. Horváth T, Gärtner T, Wrobel S (2004) Cyclic pattern kernels for predictive graph mining. Association for Computing Machinery, New York, NY, USA, KDD '04, pp 158–167
15. Kim J, Rousseau F, Vazirgiannis M (2015) Convolutional sentence kernel from word embeddings for short text categorization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp 775–780
16. Leopold E, Kindermann J (2002) Text categorization with support vector machines. How to represent texts in input space? Mach Learn 46(1–3):423–444
17. Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins C (2002) Text classification using string kernels. J Mach Learn Res 2(Feb):419–444
18. Malliaros FD, Skianis K (2015) Graph-based term weighting for text categorization. In: 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp 1473–1479
19. Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38(11):39–41
20. Nasir JA, Karim A, Tsatsaronis G, Varlamis I (2011) A knowledge-based semantic kernel for text classification. In: Grossi R, Sebastiani F, Silvestri F (eds) String Processing and Information Retrieval, Springer Berlin, Heidelberg, pp 261–266
21. Nikolentzos G, Meladianos P, Rousseau F, Stavrakas Y, Vazirgiannis M (2017) Shortest-path graph kernels for document similarity. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, pp 1890–1900
22. Pang B, Lee L (2004) A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. Association for Computational Linguistics, USA, ACL '04, p 271
23. Pang B, Lee L (2005) Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Association for Computational Linguistics, Ann Arbor, Michigan, pp 115–124
24. Ralaivola L, Swamidass SJ, Saigo H, Baldi P (2005) Graph kernels for chemical informatics. Neural Netw 18(8):1093–1110
25. Ramon J, Gärtner T (2003) Expressivity versus efficiency of graph kernels. In: Raedt LD, Washio T (eds) Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences (MGTS 2003) at the 14th European Conference on Machine Learning and 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2003), September 22 and 23, 2003, Cavtat-Dubrovnik, Croatia, pp 65–74
26. Schenker A, Last M, Bunke H, Kandel A (2003) Classification of web documents using a graph model. In: Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings., pp 240–244

27. Shanavas N, Wang H, Lin Z, Hawe G (2016) Centrality-based approach for supervised term weighting. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), pp 1261–1268
28. Shanavas N, Wang H, Lin Z, Hawe G (2016) Supervised graph-based term weighting scheme for effective text classification. In: Proceedings of the Twenty-second European conference on artificial intelligence. IOS Press, pp 1710–1711
29. Shawe-Taylor J, Cristianini N et al (2004) Kernel methods for pattern analysis. Cambridge University Press, Cambridge
30. Siolas G, d'Alché-Buc F (2000) Support vector machines based on a semantic kernel for text categorization. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, vol 5, pp 205–209
31. Srivastava S, Hovy D, Hovy E (2013) A walk-based semantically enriched tree kernel over distributed word representations. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Seattle, Washington, USA, pp 1411–1416
32. Valle K, Ozturk P (2011) Graph-based representations for text classification. India–Norway workshop on web concepts and technologies. Trondheim, Norway, pp 2363–2366
33. Vishwanathan SVN, Schraudolph NN, Kondor R, Borgwardt KM (2010) Graph kernels. J Mach Learn Res 11(Apr):1201–1242
34. Vitale D, Ferragina P, Scaiella U (2012) Classification of short texts by deploying topical annotations. In: Baeza-Yates R, de Vries AP, Zaragoza H, Cambazoglu BB, Murdock V, Lempel R, Silvestri F (eds) Advances in Information Retrieval, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 376–387
35. Wang P, Domeniconi C (2008) Building semantic kernels for text classification using wikipedia. Association for Computing Machinery, New York, NY, USA, KDD '08, pp 713–721
36. Wang T, Li W, Liu F, Hua J (2017) Sprinkled semantic diffusion kernel for word sense disambiguation. Eng Appl Artif Intell 64:43–51
37. Wang W, Do DB, Lin X (2005) Term graph model for text classification. In: Li X, Wang S, Dong ZY (eds) Advanced Data Mining and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 19–30