Graph Embedding Based on Nodes Attributes Representatives and a Graph of Words Representation

Jaume Gibert and Ernest Valveny

Centre de Visió per Computador, Universitat Autònoma de Barcelona
Edifici O Campus UAB, 08193 Bellaterra, Spain
{jgibert,ernest}@cvc.uab.es

Abstract. Although graph embedding has recently been used to extend statistical pattern recognition techniques to the graph domain, some existing embeddings are usually computationally expensive as they rely on classical graph-based operations. In this paper we present a new way to embed graphs into vector spaces by first encapsulating the information stored in the original graph under another graph representation by clustering the attributes of the graphs to be processed. This new representation makes the association of graphs to vectors an easy step by just arranging both node attributes and the adjacency matrix in the form of vectors. To test our method, we use two different databases of graphs whose nodes attributes are of different nature. A comparison with a reference method permits to show that this new embedding is better in terms of classification rates, while being much more faster.

1 Introduction

Most real-world problems do not fit under the usual data representation by means of feature vectors. Instead, structural representations are more suitable. Graph-based representations offer interesting properties in terms of binary relations between features allowing to adapt the representation to the complexity of data while vectors are constrained to the use of a predefined number of features.

However, while structured representations provide us with a complex and powerful description of the patterns under study, their own complexity makes the processing and analysis of graphs a really hard problem. Graph matching is the process that tries to discover the structural similarity of two graphs. To know more about graph matching we refer the reader to [1], a detailed survey that organizes the whole map of techniques for solving this problem.

On the other hand, many pattern recognition techniques have been developed for data represented in the form of feature vectors. The fact that vector spaces have strong and straightforward mathematical properties, both theoretical and practical, has contributed to the expansion of pattern analysis techniques for the case when patterns are represented by elements in a feature space. In order to make all these techniques available for the case of data structurally represented,

E.R. Hancock et al. (Eds.): SSPR & SPR 2010, LNCS 6218, pp. 223–232, 2010.

for instance, by using graphs, the scientific community has devoted several efforts to find out new ways of adapting them to structured data. Graph embedding is one of them. By graph embedding we understand a function that given a set of graphs G, it maps each graph in the set to an n-dimensional vector,

$$\begin{aligned} \phi : G &\rightarrow \mathbb{R}^n \\ g &\mapsto \phi(g) = (x_1, x_2, \ldots, x_n). \end{aligned} \tag{1}$$

Several graph embeddings have been proposed in the literature so far. Some of them are based on a spectral study of the adjacency matrix or on the Laplacian matrix of the graphs [2]. Other approaches are based on random walks, and particularly on quantum walks, in order to embed nodes into a vector space [3]. Finally, let us consider the embedding proposed in [4] which is based on similarity measures between graphs and a set of prototypes. The measure considered by the authors is the Graph Edit Distance. This embedding will be later explained more in detail as it will be used as a reference system in order to evaluate our approach.

However, some of these embeddings are constrained to specific classes of graphs or still rely on graph matching. The computational complexity of graph matching lies on what is known as the assignment problem. Nodes of one graph have to be identified with nodes of the other one, and such procedure has an exponential computational cost in the number of nodes of the involved graphs. This problem has been addressed, for instance, by means of best-first search techniques like the A^* algorithm or by bipartite graph matching procedures and the Hungarian method. Ideally, however, if nodes of one graph could directly be identified with nodes of the other, we would not have to face this problem and graph matching would be a problem with a straightforward solution.

In this paper we aim to propose a graph embedding avoiding the computational cost of graph matching through an intermediate meta-representation of the graphs in which nodes of a family of graphs become identifiable. We will call this meta-representation graph of words as the underlying idea is based on the well-known *bag of words* technique for document and image classification [5]. Thus, we will cluster node attributes to obtain representatives (words) of the node attributes of a set of graphs. This will lead to represent the whole set by graphs that have exactly the same number of nodes and share the same node

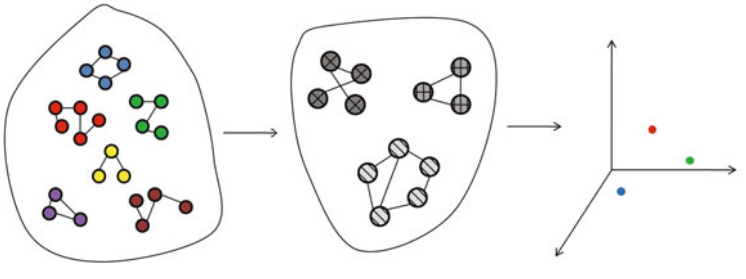

Fig. 1. General scheme of the proposed approach. First step: graph generalization. Second step: graph embedding.

labels. Then, this graph representation can be easily converted into a vector by just taking node attributes and serializing the adjacency matrix. In Fig. 1 the whole procedure is depicted.

In the remainder of this paper we describe, first in Section 2, how the generalization of a set of graphs into a graph of words can be done, and second, in Section 3, how these generalized graphs are converted into vectors. Section 4 describes the databases we have used for the experimentation part and the results and discussions are presented in Section 5. Finally, Section 6 concludes and present the future work to be done.

2 Graph of Words Representation

An attributed graph g, or just a *graph*, is a 4-tuple $g = (V, E, \mu, \nu)$, where V is a finite set of elements, called nodes, $E \subseteq V \times V$ is the set of edges, and $\mu : V \rightarrow L$ and $\nu : E \rightarrow L'$ are the corresponding labelling functions for which each node and edge is correspondingly attributed with a specific label. As there is no restriction on the set of labels for both nodes and edges, this definition allows to describe a large family of graphs. For instance, the set of nodes labels L could be represented either by the vector space $\mathbb{R}^n$ or even by a set of non-numerical attributes $L = \{l_1, l_2, l_3, \dots\}$ with a certain specific semantics. The same happens for the case of edges attributes L'. Edges of the graphs are described by pairs of nodes (u, v), where $u \in E$ is the source node and $v \in E$ is target one. Undirected graphs are all those graphs which contain both $(u, v) \in E$ and $(v, u) \in E$ satisfying $\nu(u, v) = \nu(v, u)$. Unlabelled graphs are graphs where both nodes and edges have the same label, usually called the null label ϵ. In this work we will just consider undirected graphs with unlabelled edges and with all the node labels being of the same nature.

In this section we will introduce a way to generalize almost any set of graphs described as before into a new set of graphs with the same number of nodes and sharing the same node labels. The only constraint about the input graphs is that all node attributes must be of the same type. In addition, if edges of the original graphs are labelled, these labels are ignored in the conversion procedure. The generalization is done in two steps: first, getting the set of nodes of the new graph by clustering and selecting representatives among the original node attributes and second, getting the edges that link the new set of nodes. These two steps are further described in the next subsections.

2.1 Node Representatives

As we already pointed out in the introduction, one of the main problems in graph matching is node assignment. To avoid such problem, we would appreciate an ideal situation where nodes in a family of graphs were directly in correspondence. Not by just *a priori* knowledge about the information stored in the nodes, but by the fact that those nodes were exactly representing the same attributes, in other words, that two graphs in a family of graphs share exactly the same nodes. A possible way to reach this situation is the one described as follows.

Given a set of graphs $G = \{g_1, g_2, \ldots, g_n\}$, each one with its corresponding nodes, edges and labelling functions, $g_i = (V_i, E_i, \mu_i, \nu_i)$, and the corresponding labels sets (L_i, L'_i), we want to represent the whole family of graphs using generalized representations sharing the same nodes. To do so, we first consider all node attributes $\mathcal{L} = \{L_i \,|\, i = 1, \ldots, n\}$. We assume all these sets of labels are of the same nature, this is, we do not consider -if there is such- the case in where for instance L_j would be representing numerical attributes and L_k semantic attributes.

Second, from this set of labels we select a finite number of representatives. Such representatives do not need to be elements from $\mathcal{L}$. We will hereby adopt a quite known notation extracted from the *bag of words* technique for document -originally- and image classification [5]. This technique represents an image by first extracting visual features from interest points and selecting representatives, usually called *visual words*, from the whole set of features of the images under study. The set of representatives is usually built by clustering and is called the visual vocabulary. Every interest point is assigned to its closest word in the vocabulary and finally the image is represented as a histogram of appearing words in the image. We perform the selection of node attribute representatives in a similar way. For this reason we call this new representation the *graph of words*. From the whole set of attributes $\mathcal{L}$, we select representatives, *words*, by means of any clustering technique. How the selection of representatives has to be done for a given family of graphs will depend on the nature of the nodes attributes, and it will be a key issue in the generalization graph. For instance, a k-Means algorithm could be applied if the attributes of the set of graphs belong to $\mathbb{R}^n$, but in other cases, the clustering should be done in a semantic way.

Assume we have already selected a vocabulary $\mathcal{V} = \{w_i \,|\, i = 1, \ldots, N\}$ from the set of attributes $\mathcal{L}$ of the graph family G. We call every w_i a word. The generalization of a graph $g = (V, E, \mu, \nu)$ is done as follows: every node $u \in V$ of the graph is assigned to its closest or most similar word in $\mathcal{V}$. We denote this assignment by

$$\begin{aligned} \lambda : V &\to \mathcal{V} \\ u &\mapsto \lambda(u) = w. \end{aligned} \tag{2}$$

The concept of similarity here will depend on the nature of the attributes and selected words. The set of words in the vocabulary that has been assigned to at least one point of the original graph will now constitute the set of nodes of the generalized graph. We label each node of the new graph, each word, with the frequency of nodes assigned to it. Fig. 2 shows an example of the procedure. The graph has six nodes. Three of them have been assigned to the red word, two to the green one and one to the blue word. In the generalized graph, these three words will appear as nodes, with their corresponding frequencies. In this example, we assume a vocabulary which also contains a yellow word but no node of the original graph has been assigned to it.

By this procedure we have got a representation of graphs where the set nodes of all the graphs in a given family is the same, this is, the set of representatives or vocabulary. This will allow to treat the assignment problem in a very specific

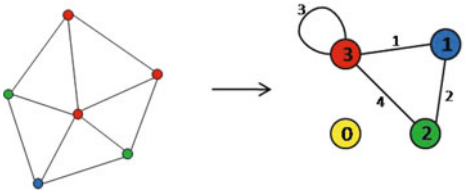

Fig. 2. A graph (left) and its generalization graph (right) or graph-of-words

way so that no computational issues are involved. However, before describing the way to deal with it, we still need to declare which is the set of edges in the graph of words representation. Next section is devoted to it.

2.2 Structural Relations of Representatives

The way of defining edges between words in the generalized graph is the most intuitive one. Given a graph g, let us assume there exist an edge $e = (u, v) \in E$. Each node is assigned to a word in the vocabulary: there are $w, w' \in \mathcal{V}$ such that $\lambda(u) = w$ and $\lambda(v) = w'$. Then we just add the edge (w, w') in the set of edges of the generalized graph. Formally,

$$(w, w') \in E' \iff (u, v) \in E \text{ such that } \lambda(u) = w \text{ and } \lambda(v) = w' \tag{3}$$

where $E' \subseteq \mathcal{V} \times \mathcal{V}$ is denoting the set of edges of the graph of words. We will label the edge $(w, w') \in E'$ with the frequency this fact occurs, this is, how many times two nodes of the original graph that have been assigned to the words w and w' are linked with an edge of the original set of edges E. Again, Fig. 2 depicts an example of the procedure. For example, a node that has been assigned to the green word is connected to a node that has been assigned to the red word. Thus, the pair $(green, red)$ will constitute an edge of the graph of words. Actually this situation occurs four times, so we label the edge with the number 4.

3 Graph of Words Embedding

Our generalized graphs, the graphs of words, are of a very particular nature as all of them share the same nodes. This property makes the transition between the domain of the graphs and a vector space a really easy problem. Next section describes formally how this can be done.

3.1 Definition

Let $g = (V, E, \mu, \nu)$ be a graph of words with respect to the vocabulary $\mathcal{V} = \{w_i \,|\, i = 1, \ldots, N\}$ of size N. Remember μ was describing the frequency of a word, and ν the frequency of a relation between two words. We can always describe the set of nodes of g as $V \equiv \mathcal{V}$. If a specific word w does not appear as

a node of the graph we can consider it as a node with $\mu(w) = 0$. Exactly as the yellow word in the example shown in Fig. 2.

The fact that we control which words are there in the vocabulary, makes the nodes of g identifiable, and therefore, sortable. We can arrange, for all graphs in a set G of graphs of words, the node labels as:

$$\begin{aligned} \phi_w^{\mathcal{V}} : G &\rightarrow \mathbb{R}^N \\ g &\mapsto \phi_w^{\mathcal{V}}(g) = (\mu(w_1), \ldots, \mu(w_N)). \end{aligned} \tag{4}$$

Let us now consider the adjacency matrix A of the graph g. Matrix A is an $N \times N$ matrix and each entry of it a_{ij} is giving the frequency of the relation between the words w_i and w_j. Here, as for the case of the nodes, we can represent a non-existing edge (w_i, w_j), by labelling it with zero value, $\nu(w_i, w_j) = 0$. This matrix is obviously symmetric since the graph-of-words is an undirected graph.

In this case, we can just arrange the adjacency matrix in the form of a vector. Not all entries of the matrix are needed since it is symmetric and that would be redundant. We can just consider, for instance, the upper part of the diagonal and the diagonal.

$$\begin{aligned} \phi_r^{\mathcal{V}} : G &\rightarrow \mathbb{R}^p \\ g &\mapsto \phi_r^{\mathcal{V}}(g) = (\nu(a_{11}), \nu(a_{12}), \ldots, \nu(a_{ij}), \ldots, \nu(a_{NN})),\ \forall\, i \leq j. \end{aligned} \tag{5}$$

Finally, we define the embedding of a set of graph of words G with respect to the vocabulary $\mathcal{V}$ as the simple concatenation of both the nodes and the edges attributes, this is, of both the words frequencies and the frequencies of the relations between words. Formally,

$$\begin{aligned} \phi_A^{\mathcal{V}} : G &\rightarrow \mathbb{R}^n \\ g &\mapsto \phi_A^{\mathcal{V}}(g) = (\phi_w^{\mathcal{V}}(g), \phi_r^{\mathcal{V}}(g)). \end{aligned} \tag{6}$$

3.2 Computational Issues and Potential Solutions

The main problem we might face after embedding the graphs using $\phi_A^{\mathcal{V}}$ is the dimension of the vectors. Indeed, such dimension increases quadratically with respect to the vocabulary size N. The first part of the vector, the words part $\phi_w^{\mathcal{V}}(g)$ has size N and the second one, $\phi_r^{\mathcal{V}}(g)$, has size $p = (N^2 + N)/2$ resulting in a vector of dimension $n = (N^2 + 3N)/2$. Processing this vector could not be treatable for a large enough vocabulary size.

Dimensionality reduction techniques could be applied in these situations. Semantically, such reduction would discover which are the specific relations of words that are really important in the graph of words representations. Also, in order to avoid such large dimensions, the whole construction of the generalized representation could be performed in a supervised manner. This is, selecting representatives class-wise and executing the embedding part of the adjacency matrix $\phi_r^{\mathcal{V}}$ by just considering relations of words belonging to the same class. We had not faced this problem as the vocabularies we have selected for our databases of graphs were not that large. The construction of the generalized graphs for two different databases is explained in the next section.

4 Experimental Setup

The whole set of experiments is carried out using two databases inspired in the IAM Graph Database Repository [6]. Even though we do not consider semantic attributes of nodes, the two situations under study refer to two different kind of attributes. In the first case, nodes are labelled with (x, y) positions and in the second one with visual descriptors.

In the way the graph of words representation has been defined, the embedded graphs clearly depend on the chosen vocabulary. For instance, the use of a small vocabulary makes easy the introduction of errors in the assignment of points to words and every word would represent too many and possibly wrong points. On the other hand, a large vocabulary would create too much sparsity on the resulting vectors making difficult further analysis of data. Due to these situations, in order to check the sensibility of the representation to the selected vocabulary, different sizes of the nodes representatives sets have been tried for each dataset.

4.1 IAM Letter Database Generalization

The graphs in this database are representing distorted letter drawings. Only the 15 capital letters of the Roman alphabet that consist of straight lines are considered: *A*, *E*, *F*, *H*, *I*, *K*, *L*, *M*, *N*, *T*, *V*, *W*, *X*, *Y*, and *Z*. For details on the construction of the graphs we refer the reader to the database reference. We use exactly the same sets as the authors in the reference: uniformly distributed graphs all over the 15 classes and a training, validation and test sets of 750 graph each. We only work on the *low* level of distortion.

By representing distorted letters using the graph of words, we aim to undo the distortion each letter has suffered. In order to build the generalized graphs for the letters, let us notice that nodes of the graphs are attributed with the (x, y) coordinates with respect to a reference coordinate system. We do not cluster all these coordinates, but instead we just pick a regular grid of $n \times n$ points in the range of the nodes attributes ($n = 3, 7, 11$). This decision was made after viewing how all nodes of all letter graphs were distributed over the plane.

4.2 COIL-100 Database Generalization

The second graph dataset we use in our experiments is representing the COIL-100 object database [7]. It consists of 100 different object, and each image is taken every 5 degrees of rotation. We represent these objects modifying the COIL-DEL representation from the IAM Graph Dataset repository. This representation takes interest points by using the Harris corner detection algorithm, labels these points with their corresponding pixel coordinates and applies a Delaunay triangulation to the nodes for the set of edges. Our modification consists of labelling such points with a visual descriptor that could provide texture information of the objects. In particular we label each node with a SIFT descriptor [8]. Again, the same sets for training, validating and testing the systems are used.

By building the generalized graphs for these objects we intent to describe the structural relations of textural patterns, this is, how visual descriptors are structurally related. In this case, we cluster the set of SIFT descriptors using the k-Means algorithm and finding $k = 50, 100, 150$ centroids as representatives. Thus, the dimensionality of the vector space where graphs are embedded is much larger than in the letter dataset, yet the vectors are still manageable.

5 Results

We want to classify both databases by using the graph of words embedding. Such classification is done by using a *Support Vector Machine* [9,10] on the embedded graphs. Kernels and parameters are properly tuned using the validation sets. We need a reference system to which we can compare how the proposed embedding works, in other words, how good is the idea of embedding the graphs by just arranging their nodes and edges attributes in a row. To do so, we select the proposed embedding in [4], by which each graph is assigned to a vector constructed out of Graph Edit Distances from itself to a set of prototypes. This embedding has been applied to the intermediate graph of words representation. Next section recalls the formal definition of the embedding.

5.1 A Reference System

Let G be a set of graphs and $\mathcal{P} = \{p_1, p_2, \ldots, p_n\} \subseteq G$ a set of prototypes. The embedding is defined as the function $\varphi_n^{\mathcal{P}} : G \to \mathbb{R}^n$ such that

$$\varphi_n^{\mathcal{P}}(g) = (d(g, p_1), d(g, p_2), \ldots, d(g, p_n)) \tag{7}$$

where $d(g, p_i)$ is the Graph Edit Distance between the graph g and the prototype p_i.

As we are using the Graph Edit Distance between instances of graph of words, we need to specify which is the cost function we have used during the matching process. Briefly, the node and edge deletion and insertion costs are the labels (frequencies) of the inserted or deleted nodes and edges. For the substitution of nodes and edges we pick the difference of labels if the substitution is of the same words or edges between the same words, and infinity otherwise.

5.2 Classification Rates

Results for both databases, the Letters and the COIL-100 objects, using the proposed embedding and the reference one are shown in Tab. 1.

We see how the proposed embedding $\phi_A^{\mathcal{V}}$ works better than the reference one no matter whether the dimension of the vectors is larger or smaller than for the graph edit distance embedding. For instance, in the letter database the embedded graphs with respect to the 3×3 grid of points live in a 54-dimensional space when using $\phi_A^{\mathcal{V}}$, and in a 750-dimensional one when the case is $\varphi_n^{\mathcal{P}}$. On the other database, when using a vocabulary size of 150 words and the proposed

Table 1. Classification rates for both databases using the proposed embedding and the reference one. Vocabularies for the Letter Database are $n \times n$ regular grids; k-Means algorithm is used for the COIL Database case. For the case of $\varphi_n^{\mathcal{P}}$ the whole set of training elements is used as the prototypes set.

Database	Vocabulary	Embedding approach	
		Proposed $\phi_A^{\mathcal{V}}$	GED-based $\varphi_n^{\mathcal{P}}$
Letter	3×3	**97.33**%	96.4%
	7×7	**99.2**%	96.13%
	11×11	**96.93**%	94.4%
COIL	50	**82.8**%	80.6%
	100	**88**%	85.6%
	150	**89.4**%	88.9%

embedding, the COIL objects are represented by really high dimensional vectors ($\simeq$ 12000 dimensions) and not that large but still (2400 dimensions) in the case of the graph edit distance based embedding.

Of course, related to what it has been already said in the introduction, the computational cost of the processing and comparison of the graphs is actually null when we arrange them in the form of vectors using $\phi_A^{\mathcal{V}}$. The other situation is not desirable since graph edit distance takes a lot of time. For example, the embedding of COIL objects using $\varphi_n^{\mathcal{P}}$ took a couple of days in a regular personal computer.

6 Conclusions and Future Work

In this work, we have proposed a new way of solving the graph classification problem by means of embedding graphs into vectorial spaces. The embedding procedure consists mainly in two steps: a generalization of the graphs using what we call the graph of words representation by assigning nodes to attribute representatives and then, arranging this new representation in the form of vectors. The experiments have shown the power of the described approach in terms of classification rates and time consuming with respect to a well-known and established embedding of graphs.

The generalization step, the so-called graph of words representation, is dependent on the choice of a set of representatives of the nodes attributes. Different selections of both vocabulary types and sizes will lead to different classifiers, not necessarily one better than others. Small vocabularies will gather more points together into every word, describing global structural relations between nodes of the original graphs, while larger vocabularies will distribute the points into more specific words retaining local information of the graphs under study. Such a situation could be exploited by ensembles of classifiers that would increase the overall accuracy of the individual classifiers.

The whole procedure explained does not take into account edge attributes of the graphs. Several graph databases do have edge attributes in their graphs because such attributes may describe important information about the relation between nodes of the graphs. It is quite clear that we should adapt our graph embedding to those cases where edge attributes are present. One possible way to proceed in this situation would be to also cluster the edge attributes and add a new semantic feature to the relation between words. Then, we should probably face the problem of dimensionality reduction.

This paper is mainly about a really preliminary work in the sense that a lot of experimentation is still needed. We focused on evaluating only the second step in the whole procedure: the representation of the graph of words using vectors. A comparison between the whole proposed embedding, which needs a transition to the generalized graph, and embeddings of the original graphs with other techniques has to be done and constitutes the current work of the authors.

References

1. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. Journal of Pattern Recognition and Artificial Intelligence 18(3), 265–298 (2004)
2. Luo, B., Wilson, R.C., Hancock, E.R.: Spectral embedding of graphs. Pattern Recognition 36(10), 2213–2230 (2003)
3. Emms, D., Wilson, R.C., Hancock, E.R.: Graph Embedding Using Quantum Commute Times. Graph-based Representations in Pattern Recognition, 371–382 (2007)
4. Bunke, H., Riesen, K.: Recent Developments in Graph Classification and Clustering using Graph Embedding Kernels. Pattern Recognition in Information Systems, 3–13 (2008)
5. Dance, C., Willamowski, J., Fan, L., Bray, C., Csurka, G.: Visual categorization with bags of keypoints. In: ECCV International Workshop on Statistical Learning in Computer Vision, pp. 1–22 (2004)
6. Riesen, K., Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
7. Nene, S., Nayar, S., Murase, H.: Columbia Object Image Library: COIL-100. Technical report, Dept. of Computer Science, Columbia University, New York (1996)
8. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60, 91–110 (2004)
9. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
10. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001) Software available at, `http://www.csie.ntu.edu.tw/~cjlin/libsvm`