2009 10th International Conference on Document Analysis and Recognition

Vector Representation of Graphs : Application to the Classification of Symbols and Letters

Nicolas Sidère[1,2] Pierre Héroux[1] Jean-Yves Ramel[2]

[1]Université de Rouen
LITIS EA 4108
BP 12
76801 Saint-Etienne du Rouvray, FRANCE

[2]Université François Rabelais Tours
Laboratoire Informatique de Tours
64 Avenue Jean Portalis
37200 Tours, FRANCE

{nicolas.sidere, ramel}@univ-tours.fr, pierre.heroux@univ-rouen.fr

Abstract

In this article we present a new approach for the classification of structured data using graphs. We suggest to solve the problem of complexity in measuring the distance between graphs by using a new graph signature. We present an extension of the vector representation based on pattern frequency, which integrates labeling information. In this paper, we compare the results achieved on public graph databases for the classification of symbols and letters using this graph signature with those obtained using the graph edit distance.

1 Introduction

One of the main interest in document analysis is the recognition of some objects, like characters or symbols. The character recognition can be used in textual documents, such as newspapers, letters or checks. The symbol recognition is more used in technical drawings or architecural plans indexation. Some works, such as [2] or [7], are focused on these domains.

Our article focuses on the classification of graphs representing, in our case, these objects. With their representative power, graphs are more and more used in pattern recognition. This rise-up is also due to the increase of computation power possibilities. In fact, the pattern recognition is often based-on the computation of a distance between two graphs. The lower the distance is, the more the graphs can be considered as similar. The computation of this distance is NP-Complete [6]. Several works on this topic are presented in [3]. Therefore, many dedicated algorithms with reducing the computation needs to search the most similar graph or subgraphs, for example [12] or [3].

An interesting approach is to embed a part of the topology of a graph into a vector feature space, for example a numerical vector. This representation reduces the computation of the distance between two graphs to the computation of a distance between two vectors, ie. to a linear complexity; but it needs a complex indexing time which has to be done during an off-line period. We pointed out two interesting works from the literature :

- A first method is presented in [9]. This representation is based on local descriptors, which are the vertex degree and the labels of connected edges. The simplicity of the description combined with a comparison of two graphs reduced to a linear time allows to find similar graphs in most of cases.
- In the second approach presented in [1] the representation of a document is focused on the extraction of frequent subgraphs. The structural representation of a document is sum up by counting the occurences of a lexicon built with frequent subgraphs. This lexicon of frequent subgraphs vary from a database to another.

The methods of embedding a graph in a vector presents advantages, but also drawbacks. The first problem is the bijectivity between the graph space and the vector space. In fact, this is the consequence of the loss of informations induced by the extraction of the vectorial representation. The descriptors described in [9] can induce some perturbations because of the very local nature of the approach. The second method [1] appears to be more expressive than the first one because of descriptors that are more global and embed more topological information. But, whereas the

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.218

IEEE computer society

first method is very generic and can be applied on every databases, the second one needs a rich knowledge of the domain and does not allow to process a set of document with a huge heterogeneity. Thus, the idea of improving the two works has grown with the wish to associate a less local vectorial description with a much more generic vectorial representation.

Our approach relies on these works. We presented in [11] a new vectorial description based on a lexicon which embeds some rich topological informations as in [1] with keeping the interesting genericity of [9]. The first experiments were made on unlabeled graphs. Even if they were comparable to the literature, the results highligted the need to embed the label and encouraged us to improve our vectorial representation in order to include more informations than topology in the description. This paper talks about the evolution of our vectorial description.

The next section presents our lexicon and the vectorial representation which now embeds the labeling information. Section 3 presents the first results leaded on symbols and letters. Finally section 4 concludes the paper and proposes some extensions.

2 The vectorial representation

As said before, the lexicon is the basement of the construction of our graph vectorial signature. So, the lexicon content is quite determinant in the relevance of the vectorial representation. As we want the representation scheme to be as generic as possible, it is preferred to use a lexicon independent from the database content. However, this lexicon must be comprehensive enough to ensure that it allows to discriminate a graph from another.

As we want to associate the advantages of the two methods presented in the introduction, we have therefore decided to take as a baseline the non-isomorphic graphs network presented in [8]. The network presents all graphs composed of n edges up to N (where N is the maximum number of edges). This network is built iteratively from a graph pattern made up of a single vertex. At each iteration, it is possible to build a pattern of rank n by adding an edge to a pattern of rank $n-1$ with the ability to add a vertex if needed. All solutions are considered. This results in a network complete. A pattern with rank n built from a pattern with rank $n-1$ is called successor. Conversely, the pattern of $n-1$ is called predecessor. A pattern of this network may have several successors. Similarly, several patterns with rank $n-1$ can rise to the same successor. Ways of construction of this non-isomorphic graph network can be stored to build all predecessors and successors of a graph.

The lexicon is composed of all patterns until a maximal rank. Thereafter, the term *pattern* will refer to a subgraph element of the non-isomorphic graph network.

For example, Figure 1 shows the non-isomorphic graph network until the fourth rank giving a lexicon of 11 patterns. The dotted arrows indicate the paths of construction of the network, the arrows are directed from the predecessors towards the successors.

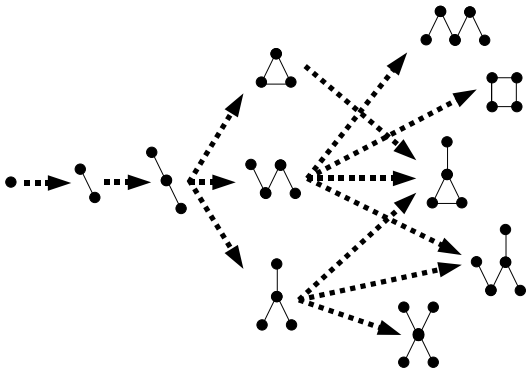

Figure 1. The non-isomorphic graph network

Table 1 gives the number of elements in the lexicon depending on the maximum rank of the non-isomorphic graph network.

Rank	Size
0	1
1	2
2	3
3	6
4	11
5	23
6	51
7	117
8	276

Table 1. Size of the lexicon depending on the rank of the non-isomorphic graph network

We can notice that the number of patterns increases exponentially with the rank. The size of the lexicon is a parameter to determine according to several criteria. Indeed, the complexity of the transformation to a vectorial representation is directly dependent of the number of patterns. However, the more the size of the lexicon increases, the bigger the integrated patterns are. The vectorial representation then integrates more information on topology. Therefore, it is necessary to find a trade-off between expressiveness and complexity.

The vectorial representation of a graph topology will be built by a count of the occurences of each pattern of the lexicon. In other way, each element of the vector is the frequency of apparition of a pattern, which represents a descriptor of a part of the graph. Thus, the topology of the

graph is embedded in the vectorial representation. Figure 2 shows a simple graph and the table 2 the topology-based vectorial representation.

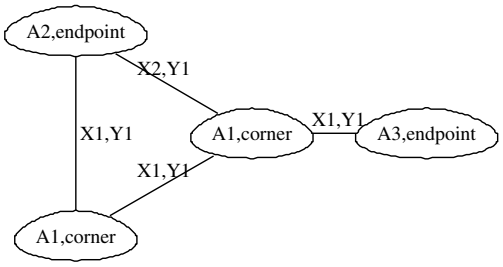

Figure 2. A simple labeled graph

Pattern	•	—•	—•—	◺	—•—•	⅄
Freq.	4	4	4	2	1	1

Table 2. Topology-based vectorial representation of the graph in fig.2

At this point, the vectorial representation embeds only some topological information and needs to be enriched by encapsulating the labels of the edges and vertices. We decided to work on multi-labeled graphs, i.e. graphs with labels on vertices and nodes. Each of these labels can be composed with several attributes. The inclusion of these labels is done in four steps :

1. The first step is to list all the labels which occur at least once in the database, for the vertices and for the nodes. At the end of this step, there are as many lists as the number of types of labels.
2. The second step is to discretize numerical labels. As the vector is based on the frequency of appearance of patterns, only nominal labels are considered. The discretization is done the simplest way by splitting the numerical interval n classes. Then, the new labels are affected to the graph.
3. The third step is the computation of all possible combinations of labels for a vertex if it is caracterized by several attributes. The same is done for edges.
4. The fourth step is to affect to each topological pattern a vector of possible vertices and edges.

The lexicon which was a vector in [11] can now be considered as a table. Each column corresponds to a pattern from the lexicon and is then relative to the topological information. Each line of the table is relative to a label combination.

Pattern	•	—•	—•—	◺	—•—•	⅄
Freq.	4	4	4	2	1	1
A1, corner	2	4	7	4	2	2
A2, corner	0	0	0	0	0	0
A3, corner	0	0	0	0	0	0
A1, endpoint	0	0	0	0	0	0
A2, endpoint	1	2	3	2	1	1
A3, endpoint	1	1	2	2	0	1
X1, Y1	0	3	6	5	2	2
X1, Y2	0	0	0	0	0	0
X2, Y1	0	1	2	1	1	1
X2, Y2	0	0	0	0	0	0

Table 3. Vectorial representation of the graph in fig.2

The construction of the vectorial representation can now be performed. The construction of the represetation consists on filling all the cells of the table generated with the frequency of each pattern and each label for this pattern.

Table 3 presents an example of the proposed vectorial representation for the graph represented on Fig. 2. We consider that each vertex is labeled with two attributes A and $Type$ such as $A = \{A1, A2, A3\}$ and $Type = \{Endpoint, Corner\}$. Edges also have two attributes X and Y with $X = \{X1, X2\}$ and $Y = \{Y1, Y2\}$.

3 Experiments

This section deals with some experiments we conducted through two databases. We use the vectorial representation to classify symbols and letters. We leaded the work on datas available at this url `http://www.iam.unibe.ch/fki/databases/iam-graph-database` which are public. Some results are given in [10]. Each of the datasets is divided in three disjoint subsets ie. training, validating and testing. In order to benchmark our method, we compare our results. For each dataset, the classification result of a k-nearest neighbor classifier (k-NN) used with graph edit distance which is the reference, and with the vector description. First, the classifier is trained and validated on two different subsets, in order to reach the optimal k. The results which are presented are claimed on the third subset. The same process is applied to the edit distance and the vector representation. The two next subsections present the two databases (as they are introduced in [10]) and results of the classification tests, in order to conclude in the last subsection.

3.1 Letter database

This graph data set involves graphs built from 15 capital letters of the Roman alphabet (A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z). A prototype is drawn for each letter and converted into a graph. Lines are represented by undirected edges which are unlabeled. Vertices, which consider ending points of the drawing, are labeled with a two-dimensional attribute giving its position relative to a reference coordinate system. 2250 graphs builds the data set. They are uniformly distributed over the 15 classes. In order to test classifiers under different conditions, distortions are applied on the prototype graphs with three different levels of strength, low, medium and high. Hence, the total data set comprises 6,750 graphs altogether. Instances of the letter A and its distortion models are represented in Fig. 3. The authors of [10] achieved classification rates of 99.6% (low), 94.0% (medium), and 90.0% (high). The following results are based on the low distortion database.

Figure 3. Examples of differents distortion of the letter A

	2	3	4
3	86.26%	91.46%	87.59%
6	85.18%	91.86%	89.30%
11	85.60%	91.33%	88.90%
23	84.91%	91.07%	88.79%

Table 4. Recognition rates for letter database

Table 4 shows several rates achieved with different size of the lexicon (3, 6, 11 or 23 patterns) and different discretization (the space is separated in 2, 3 or 4). We notice that the best results are obtained with a cut by 3 of numerical attributes. Even if, for this parameter, all the rates are quite close, the extraction with the lexicon of 6 patterns (graphs with maximum 3 edges) is a little bit better. After, the recognition rate falls slowly. The results are 8% lower than the reference. This difference is the consequence of the noises in the graphs induced by the distortion applied to the letters. As the topology of the graph takes an important an important place in our representation, the distortion of the letters impacts on the results. To make our representation more robust, we work on an evolution which will take into an account the redundancy induced by the network construction of the lexicon (cf. fig.1).

3.2 GREC database

This data set consists of graphs built from the GREC database. This database is composed of symbols extracted from architectural and electronic drawings. Different distortion, morpholigical operations (like erosion or dilatation)are applied. Then, a skeletization process is applied to obtain a single piwel wid line. Finally, graphs are extracted from the resulting denoised images by tracing the lines from end to end and detecting intersections as well as corners. Vertices can be either ending points, corners, intersections or circles; they are also labeled with their position by two numerical attributes. Undirected edges connects the vertices. They can be labeled as line with an angle with respect to the horizontal direction or as arc with the diameter. From the original GREC database [4], 22 classes are considered. For an adequately sized set, all graphs are distorted nine times to obtain a data set containing 1,100 graphs uniformely distributed over the 22 classes. The resulting set is split into a traininig and a validation set of size 286 each, and a test set of size 528. The classification rate achieved by the author of [10] on this data set is 95.5%.

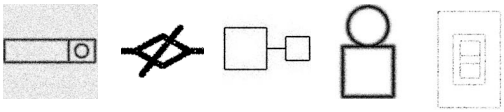

Figure 4. Examples of GREC symbols

	2	3	4
3	94.50%	94.05%	94.34%
6	94.71%	95.83%	93.93%
11	94.89%	95.83%	94.69%
23	95.26%	95.83%	94.69%

Table 5. Recognition rates for GREC database

The table 5 is built the same way as table 4. Many parameters were tested : number of patterns in rows and number of discretized classes in columns. The best rate achieved is a little higher than the reference, and reached for 6 patterns and numerical attributes discretized in 3 classes.

3.3 Conclusion

In this section, we presented different results of classification with different parameters, which must be adapted to the database. In our cases, the size of lexicon and the number of classes in discretization that are better, seems to be the same. The optimal size of the lexicon depends on

the size (mean of nodes and edges) of the graphs and the best choice for the number of classes depends on attributes. We are thinking about applying an algorithm like in [5] to discretize numerical datas to improve the recognition rates.

4 Conclusions and extensions

This article has presented a new method to embed graphs into a vector representation. We showed some results that proves that the representation can be efficient depending on the choice of parameters. The genericity of the approach, due to the independance of the lexicon with the datas, is also shown in this article. With the same extraction, the classification rates reached are comparable to the literature.

But the reduce of the complexity of the computation of a distance between two graphs is an interesting property. In a case of symbol or letter retrieval, where a user want to query a database with an example and get in return similar objects, the computation time of the distance is more critical than in classification task. Even if the extraction of the vector representation is NP-Complete, the database can be indexed off-line. Then, the vector extracted from the query of the user is compared with all the vectorial representations of the graphs in the database in a linear time. Our future work will consist on showing the gain of time in this case of use.

Acknowledgments

The works described in this article were done with the support of the ANR within the NAVIDOMASS ANR-06-MDCA-12 project.

References

[1] E. Barbu, P. Héroux, S. Adam, and E. Trupin. Clustering document images using a bag of symbols representation. In *ICDAR*, pages 1216–1220, 2005.

[2] A. K. Chhabra. Graphic symbol recognition: An overview. In *GREC '97: Selected Papers from the Second International Workshop on Graphics Recognition, Algorithms and Systems*, pages 68–79, London, UK, 1998. Springer-Verlag.

[3] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 2004.

[4] P. Dosch and E. Valveny. Report on the second symbol recognition contest. In *GREC*, pages 381–397, 2005.

[5] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. pages 1022–1027, 1993.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[7] V. K. Govindan and A. P. Shivaprasad. Character recognition—a review. *Pattern Recogn.*, 23(7):671–683, 1990.

[8] J. Jaromczyk and G. Toussaint. Relative neighborhood graphs and their relatives. In *Proceedings of the IEEE*, 1992.

[9] D. Lopresti and G. Wilfong. A fast technique for comparing graph representations with applications to performance evaluation. *Int. J. Doc. Anal. Recognit.*, 6(4):219–229, 2003.

[10] K. Riesen and H. Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *SSPR/SPR*, pages 287–297, 2008.

[11] N. Sidère, P. Héroux, and J.-Y. Ramel. A vectorial representation for the indexation of structural informations. In *SSPR/SPR*, pages 45–54, 2008.

[12] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.