Combination of Product Graph and Random Walk Kernel for Symbol Spotting in Graphical Documents

Anjan Dutta†, Jaume Gibert†, Josep Lladós†, Horst Bunke†† and Umapada Pal†††
†*Computer Vision Center, Universitat Autònoma de Barcelona, Barcelona, Spain*
††*Institute of Computer Science and Applied Mathematics, Universitat Bern, Bern, Switzerland*
†††*Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India*
{*adutta, jgibert, josep*}*@cvc.uab.es, bunke@iam.unibe.ch, umapada@isical.ac.in*

Abstract

This paper explores the utilization of product graph for spotting symbols on graphical documents. Product graph is intended to find the candidate subgraphs or components in the input graph containing the paths similar to the query graph. The acute angle between two edges and their length ratio are considered as the node labels. In a second step, each of the candidate subgraphs in the input graph is assigned with a distance measure computed by a random walk kernel. Actually it is the minimum of the distances of the component to all the components of the model graph. This distance measure is then used to eliminate dissimilar components. The remaining neighboring components are grouped and the grouped zone is considered as a retrieval zone of a symbol similar to the queried one. The entire method works online, i.e., it doesn't need any preprocessing step. The present paper reports the initial results of the method, which are very encouraging.

1. Introduction

Symbol spotting can be defined as locating a given query symbol into a set of graphical document images. The desired output for a particular query should be a ranked list of retrieved symbols in which the true positives should appear at the beginning. In document analysis it is applied for document retrieval and indexing tasks. Since documents often suffer from various noise and distortion symbol spotting is a difficult problem. Spotting is usually done through query by example *i.e.* the user crops the item he wants to retrieve from the database and the cropped image acts as input of the system. This implies an infinite number of possibilities of query symbols, which prevents explicit training of the spotting system.

The list of approaches proposed for spotting symbols in graphical documents is long [12]. The current paper only contains the key works of symbol spotting dealing with the graph representations of graphical documents, as they are more related to this work. The algorithms proposed by Messmer and Bunke [8] and Lladós *et al.* [5] were among the first few approaches of symbol spotting using graph representation. More recently, Nayef and Breuel [9] proposed a branch and bound algorithm for spotting symbols in documents, where they used geometric primitives of images as features. Luqman *et al.* [6, 7] also proposed a graph embedding based symbol spotting method, where the candidate regions containing symbols are filtered out before hand using some criteria. Recently Dutta *et al.* [2] proposed graph factorization based symbol spotting methods for architectural floorplans. Attributed graphs provide a robust representation of graphical documents and symbol spotting can be seen as a subgraph matching problem. But the major problem with most of the graph based methods is their high computational complexity. Moreover some of the methods for spotting symbol use an offline preprocessing or indexation step which adds an additional overhead. Also at the same time this offline step restricts the flexibility of the method to apply on any unindexed documents. The above reasons motivate us to propose a new method where we combine product graph and random walk graph kernel for spotting symbol on the fly without any preprocessing or offline step.

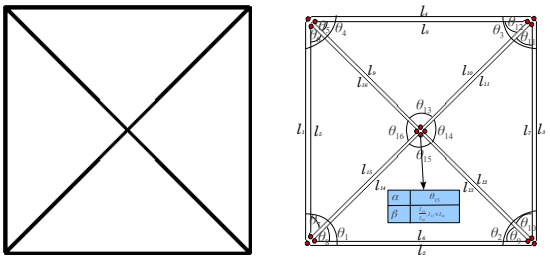
The graph representing the model or query symbol is referred to as the model graph G_M and the graph representing the document as input graph G_I. We compute the product graph G_P of $G_I(V_I, E_I, \alpha, \beta)$ and $G_M(V_M, E_M, \alpha, \beta)$, where V_I, V_M are the sets of nodes

and E_I, E_M are the sets of edges of the graphs G_I, G_M respectively and α, β are two node labeling functions to be discussed in the subsection 2.1. In general, the adjacency matrix A_P of G_P gives the number of common similar paths of length one in G_M and G_I; similarly A_P^n gives the number of common similar paths of length n between them. Product graphs are mainly been used for computing graph kernels [3]. In this work we use it for getting the candidate components or subgraphs in G_I having similarity with the components or subgraphs in G_M. Computing product graphs is an easy and comparatively efficient procedure and this gives us the opportunity to compute it online. In this work the walk kernel [4] is used to calculate the distance between the components or subgraphs, this distance value later in the second step being used to filter out the candidate components which are dissimilar from the components of G_M. This two layered approach is proposed to maintain a better trade-off between the performance and computational complexity.

The rest of the paper is organized into three sections. In Section 2 we present the methodology to represent a database in terms of the descriptors of graph paths. Section 3 contains the detailed experimental results. After that, in Section 4, we conclude the paper and discuss future directions of work.

2. Methodology

Our attributed graph representation considers the critical points detected by a vectorization method as the nodes and the lines joining them as the edges. We vectorize the contour of the line segments as reported in [11]. Note that the vectorization only results in closed loops or open edges (see Figure 1). This further ensures that each of the nodes is connected with only two adjacent nodes *i.e.* the maximum degree of any node is two.



(a) (b)

Figure 1. (a) An example symbol of table, (b) The graph representation of the vectorized document.

2.1. Node labels

Two functions α, β are used to assign labels to nodes. Note that our graph representation does not include edge labels. The two node labeling functions include the edge and angle information together. We define the function $\alpha : V \rightarrow L_{V_\alpha}$ as

$$\alpha(v) = \begin{cases} acute\angle v, & v \; is \; a \; node \; of \; degree \; two \\ 0, & otherwise \end{cases} \quad (1)$$

and node labeling function $\beta : V \rightarrow L_{V_\beta}$ is defined as

$$\beta(v) = \begin{cases} \frac{l_1}{l_2}, & v \; is \; a \; node \; of \; degree \; two \\ 0, & otherwise \end{cases} \quad (2)$$

where l_1, l_2 are the lengths of the two connecting edges to the vertex v with $l_1 \leq l_2$. The same node labels are used to compute both the product graph and the random walk graph kernel.

2.2. Product graph

Product graph $G_P = (V_P, E_P)$ of G_I and G_M relates the input graph G_I and the model graph G_M with the following properties:

- The vertex set $V_P \subseteq V_M \times V_I$ is defined by pairs of nodes with similar labels, *i.e.*,

$$V_P = \{(u_M, u_I) \,|\, u_M \in V_M,\, u_I \in V_I, \\ \alpha(u_M) \simeq \alpha(u_I) \text{ and } \beta(u_M) \simeq \beta(u_I)\}$$

- Moreover, two vertices (u_M, u_I) and (v_M, v_I) of G_P are connected by an edge if and only if

$$(u_M, v_M) \in E_M \text{ and } (u_I, v_I) \in E_I.$$

We use the parameters t_α, t_β for regulating node label similarities: $\alpha(u_M) \simeq \alpha(u_I) \Leftrightarrow |\alpha(u_M) - \alpha(u_I)| \leq t_\alpha$ and $\beta(u_M) \simeq \beta(u_I) \Leftrightarrow |\beta(u_M) - \beta(u_I)| \leq t_\beta$.

From the above definition of product graph it is clear that any path in a product graph corresponds to similar paths in both G_I and G_M. We utilize this property of product graph to get candidate components having similar paths. Let us denote the set of candidate components in G_I as C_{G_I} and all the components in G_M as C_{G_M}. In the second step, each of the components in C_{G_I} is assigned a distance value, which is the minimum of the distances of a single component in C_{G_I} with all the components in C_{G_M}. This distance is used for eliminating the components having distance value greater than a threshold t_1.

2.3. Random walk kernel

To measure the similarity between two subgraph components we define simple random walk based graph kernel. Traditional graph matching methods compute this in terms of node and edge similarities, for example, with graph edit distance. But this kind of computation is costly. Graph kernels are very suitable because they allow one to compute distances in terms of an implicitly embedded feature space. Basically, the kernel function $\kappa(x, y)$ measures the similarity of objects x and y; the larger the value of κ is, the greater is the similarity of the objects x and y.

Let us introduce the basic random walk graph kernel in a formal way. Let $g_1 = (V_1, E_1, \alpha, \beta)$ and $g_2 = (V_2, E_2, \alpha, \beta)$ be two graphs. For completeness, we again compute the product graph in a general form. The product graph of g_1 and g_2 is a graph $g_x = (V_x, E_x)$ with

$$V_x = \{(x, y) \,|\, x \in V_1,\, y \in V_2,\, \alpha(x) \simeq \alpha(y),\, \beta(x) \simeq \beta(y)\}$$

$$E_x = \{(e_1, e_2) | e_1 \in E_1, e_2 \in E_2\}$$

Let A_x be the adjacency matrix of g_x. A_x is a binary matrix of dimension $|V_x| \times |V_x|$ with $A_x(i, j) = 1$ if and only if $x_i, x_j \in V_x$ and $(x_i, x_j) \in E_x$. It is easy to verify that there exists a walk with some label sequence in g_x if and only if a walk with same label sequence exists in both g_1 and g_2. Therefore, in order to find common walks in g_1 and g_2, one can construct the product graph g_x first and then extract walks from g_x. This leads to the equation:

$$\kappa(g_1, g_2) = \sum_{i,j=1}^{|V_x|} \left[\sum_{n=0}^{\infty} \lambda_n A_x^n \right] \tag{3}$$

We multiply the adjacency matrix A_x n-times with itself, weight the product with λ_n $(n = 0, 1, ...)$ and sum all the matrices obtained this way. Finally, we sum the individual matrix elements to obtain the value of the kernel function. Actually, for any adjacency matrix A_x, matrix A_x^n gives the number of different walks of length n between any two nodes. Therefore, $\kappa(g_1, g_2)$ is in fact the number of common walks of infinite lengths, where each walk of length m is weighted by λ_m. In order to compute the infinite sum, one utilizes the fact that for $\gamma \geq 0, \gamma \in \mathbb{R}$, $\lim_{n\to\infty} \sum_{i=0}^{n} \gamma^i A_x^i$ exists, if $\gamma \leq \frac{1}{a}$, where $a =$ minimum degree found in g_1 or g_2. So the following equation holds:

$$\lim_{n\to\infty} \sum_{i=0}^{n} \gamma^i A_x^i = (\mathbb{I} - \gamma A_x)^{-1}$$

which turns equation (3) into:

$$\kappa(g_1, g_2) = \sum_{i,j=1}^{|V_P|} (\mathbb{I} - \gamma A_x)^{-1} \tag{4}$$

Finally the distance between two graph components is computed by the Euclidean distance in the implicit feature space of the kernel function:

$$d(g_1, g_2) = \sqrt{\kappa(g_1, g_1) - 2\kappa(g_1, g_2) + \kappa(g_2, g_2)} \tag{5}$$

In our case one of the two graph operands of eqn. (5) comes from C_{G_I} and the other from C_{G_M}. Each of the elements in C_{G_I} is assigned with the minimum of the distances to all the components of C_{G_M}.

3. Experimental results

We have evaluated the performance of our method on the SESYD (floorplans)[1] database which is a synthetically generated graphical document benchmark [1]. Actually, this dataset contains 10 different subdatasets, each of which consists of 100 different synthetically generated floorplans and 16 model symbols. All the floorplans in a subdatasets are created on a same floorplan template by putting different model symbols in different places in random orientation and scale. The query symbol is always ideal and does not contain any distortion. The average number nodes in the query graph and the input graph are 12 and 1500 respectively. Since we are focused on the document retrieval aspect of the problem, we use the standard performance measures of precision (**P**), recall (**R**) and F-measure (**F**) for evaluating the performance of our system. For a more detailed discussion on performance evaluation of spotting systems we refer to [10].

The influence of both parameters t_α,t_β is similar. Increasing their values makes the product graph larger which increases the recall value but reduces the precision normally reduces the F-measure. As the size of the product graph increases, the computation time also increases. For all the following experiments we set $t_\alpha = 1$, $t_\beta = 0.05$, $t_1 = 0.7$ and $\gamma = 0.1$. All the parameters are chosen experimentally. Showing the details selection procedures of the parameters is not possible due to the page limitation. Unless otherwise stated, paths of length two are considered to compute the product graph as this configuration produced the best results.

In this paper we present initial results of combining the product graph with the random walk based graph kernel. The precision and recall values of the retrieval only using the product graph are 19.4% and 92.23%, respectively, while the F-measure is 29.64%. The introduction of the walk kernel with product graph improves the precision value to 38.11%, compromises the recall to 60.08% which altogether improves the F-measure to 37.23%. The average time taken for spotting a query

[1] http://mathieu.delalandre.free.fr/projects/sesyd/index.html

Table 1. Results with SESYD dataset

Path length	P	R	F
1	24.63	80.22	33.79
2	38.11	60.08	37.23
3	26.35	77.98	36.85
4	32.76	60.08	36.56

symbol in a floorplan is 0.95 sec (approx.) which proves the method to be efficient for spotting symbol.

The next experiment is done to show how adding more path information effects the results. Here we experimented with four different path lengths. From Table 1, its clear that using paths of length two improves the performance compared to path length one. This is justified since paths of length two add more information and structural discrimination. It's also interesting to observe that increasing the path lengths after that does not improve the results. This is due to the tottering effect which reduces some discrimination.

The results shown in the experiments are still below the state of the art method [2] but it seems encouraging to work with the current strategy for a number of reasons. The method provides good recall values which means that the method is able to capture most of the instances, but it shows lower precision, due to the presence of false positives. This happens since the method works on independent components. Hence, it fails to capture the global topological structure of a symbol *i.e.* how the independent components are aligned in a symbol. Modeling individual symbols in terms of the organization of those components is expected to eliminate those false positives and hence improve the precision.

4. Conclusions

In this paper we have proposed a combination of product graph and random walk graph kernel for symbol spotting on graphical documents. A two layer architecture is proposed for efficient performance of the system, as product graph is used to capture the candidate components, and later a random walk graph kernel is utilized to measure the similarity of the candidate components. Dissimilar components are eliminated using a threshold. Here, only using the graph kernel consumes more time and introduction of product graph helps to reduce the computational complexity. The method can be executed online with moderate computation time and does not need any preprocessing or indexing step. As we have discussed in the section 3, there are many possibilities to enhance the present method. One of them is to incorporate the independent components in a proximity graph, to have the global spatial information of the symbol included. The other interesting way could be trying different kernels or more robust node labels.

5. Acknowledgement

This work has been partially supported by the Spanish projects TIN2009-14633-C03-03, TIN2008-04998, CSD2007-00018 and the PhD scholarship 2011FI_B01022.

References

[1] M. Delalandre, T. Pridmore, E. Valveny, H. Locteau, and E. Trupin. *Building Synthetic Graphical Documents for Performance Evaluation*, pages 288–298. Springer-Verlag, Berlin, Heidelberg, 2008.

[2] A. Dutta, J. Lladós, and U. Pal. Symbol spotting in line drawings through graph paths hashing. In *Proceedings of 11th ICDAR*, pages 982–986, 2011.

[3] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of 16th COLT*, pages 129–143, 2003.

[4] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of 20th ICML*, pages 321–328. AAAI Press, 2003.

[5] J. Lladós, E. Martí, and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE TPAMI*, 23:1137–1143, 2001.

[6] M. Luqman, T. Brouard, J.-Y. Ramel, and J. Lladós. A content spotting system for line drawing graphic document images. In *Proceedings of 20th ICPR*, pages 3420–3423, 2010.

[7] M. Luqman, J. Ramel, J. Llados, and T. Brouard. Subgraph spotting through explicit graph embedding: An application to content spotting in graphic document images. In *Proceedings of 11th ICDAR*, pages 870–874, 2011.

[8] B. Messmer and H. Bunke. Automatic learning and recognition of graphical symbols in engineering drawings. In R. Kasturi and K. Tombre, editors, *Graphics Recognition Methods and Applications*, volume 1072 of *LNCS*, pages 123–134. Springer Berlin / Heidelberg, 1996.

[9] N. Nayef and T. M. Breuel. A branch and bound algorithm for graphical symbol recognition in document images. In *Proceedings of Ninth IAPR International Workshop on DAS*, pages 543–546, 2010.

[10] M. Rusiñol and J. Lladós. A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. *IJDAR*, 12(2):83–96, 2009.

[11] M. Rusiñol, J. Lladós, and G. Sánchez. Symbol spotting in vectorized technical drawings through a lookup table of region strings. *PAA*, 13:1–11, 2009.

[12] K. Tombre and B. Lamiroy. Pattern recognition methods for querying and browsing technical documentation. In *13th CIARP* LNCS, vol. 5197, Springer-Verlag, 2008.