Pattern Recognition Letters 112 (2018) 353–360

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Embedding the node-to-node mappings to learn the Graph edit distance parameters

Shaima Algabli, Francesc Serratosa*

Universitat Rovira i Virgili, Av. Països Catalans 26, Tarragona, Catalonia 43007, Spain

ARTICLE INFO

Article history:
Received 9 April 2018
Available online 20 August 2018

ABSTRACT

This paper presents a learning method to automatically deduce the insertion, deletion and substitution costs of the Graph edit distance. The method is based on embedding the ground-truth node-to-node mappings into a Euclidean space and learning the edit costs through the hyperplane that splits the nodes into mapped ones and non-mapped ones in this new space. In this way, the algorithm does not need to compute any graph matching process, which is the main drawback of other methods due to its intrinsic exponential computational complexity. Nevertheless, our learning method has two main restrictions: 1) the insertion and deletion edit costs have to be constants; 2) the substitution edit costs have to be represented as inner products of two vectors. One vector represents certain weights and the other vector represents the distances between attributes. Experimental validation shows that the matching accuracy of this method outperforms the current methods. Furthermore, there is a significant reduction in the runtime in the learning process.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

Attributed relational graphs are commonly used as abstract representations for common structures such as documents, images or chemical compounds, among others [1]. Nodes of graphs represent local parts of the object and edges represent the relations between these local parts. If we want to know the similarity between two elements, we need to apply error-tolerant graph matching techniques [3,4].

Error-tolerant graph matching techniques are based on finding a mapping between nodes so that both graphs look similar when their nodes are mapped according to this node-to-node mapping. One of the most used frameworks to concretise the error-tolerant graph matching is through the Graph edit distance [5,6]. The main idea is to define the difference between graphs as the amount of distortion required to transform one graph into another through substituting, deleting or inserting nodes and edges. To do so, some penalty costs are defined for these edit operations. In this paper, we present an automatic method to learn these costs. The aim is to obtain these values automatically and therefore, in a recognition process, any error-tolerant graph matching algorithm can be computed having these costs as input parameters that have been found trough an optimisation process.

Moreover, in some object retrieval applications, in which elements are represented by graphs, the aim is to deduce which are the most similar graphs, without the graphs being previously classified. In these cases, it is crucial the method for learning is designed to learn the edit costs such that the best node-to-node mapping between pairs of graphs is computed instead of maximising the classification ratio. This is the reason why the whole process we present is dependent on a ground-truth node-to-node matching. Recently, a graph database generator has been presented that returns pairs of graphs with their ground truth correspondence [7].

Fig. 1 shows the basic scheme of our method. Given an initial database with some ground-truth node-to-node mappings, the mapped and deleted nodes are embedded in a Euclidean space and then a hyper plane is learned that splits both types of nodes (mapped and non-mapped). Finally, the costs are deduced given this hyper plane.

The outline of the paper is as follows. In the next section, we review the state of the art related to learning the edit costs. In Section 3, we define attributed graphs and Graph edit distance. In Section 4, we present our learning strategy. In Section 5, we show the experimental validation and finally, we conclude the article in Section 6.

* Corresponding author.
E-mail address: francesc.serratosa@urv.cat (F. Serratosa).

https://doi.org/10.1016/j.patrec.2018.08.026
0167-8655/© 2018 Elsevier B.V. All rights reserved.

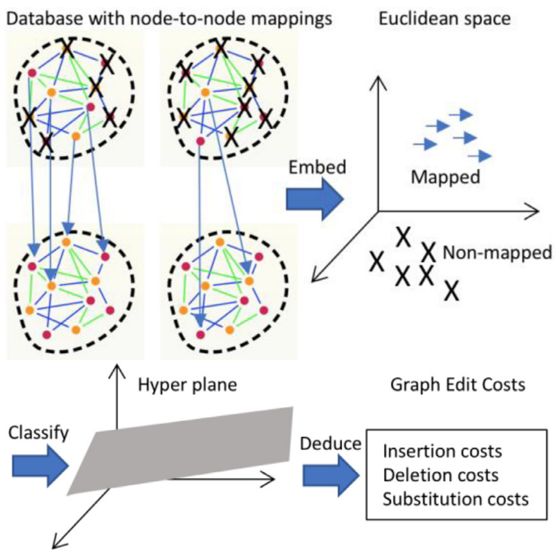

Fig. 1. Basic scheme of our learning method.

2. Literature review

The aim of this paper is to present a new method to learn the parameters of the GED. To our knowledge, only eight papers have been published related to learning these parameters: [8–14]. An important feature of these methods is the type of costs the learning algorithm obtains. The method in [8] obtains a self-organising map and the method in [9] deduces a probability density function. Therefore, in these cases, classical graph matching algorithms have to be adapted to be applied to these learning methods since these matching algorithms assume edit costs are real numbers. In [12], the human (or another system) interacts with the automatically obtained matching between nodes and imposes a new node-to-node mapping. Then, the method considers this new mapping and updates the whole correspondence between graph nodes.

Methods [10,11] and [13] assume nodes and edges have several attributes (for instance features obtained by SIFT descriptors) and these methods obtain a weight for each individual feature. However, they do not learn the insertion and deletion costs. There are some graph databases [27] whose nodes and edges have only one attribute or they are unattributed. In these cases, it makes no sense to learn the substitution weights for each feature as in [10,11] and [13], but it is crucial to learn the best combinations of insertion and deletion costs on nodes and edges as unique real numbers. In these cases, the method presented in [14] is the only one in the literature that learn the insertion and deletion edit cost on nodes and edges as constants (a real number).

The method in [14] deduces these costs through minimising the distance between the cost of a ground-truth matching and the cost of the automatically deduced matching. The method has the drawback that the learning process is very slow since the matching between each pair of graphs has to be computed in each iteration of the minimisation algorithm. Moreover, the algorithm depends on some initial parameters.

In this paper, we present for the first time, a method that learns, all at once, the insertion and deletion edit costs (as method in [14] does) and also the weights on the substitution costs (as methods in [10,11] and [13] do).

3. Graphs and Graph edit distance (GED)

In this section, we define attributed graphs, error tolerant graph matching and the Graph Edit Distance (GED). Moreover, we show a usual approximation of the GED.

Attributed graphs.

An attributed graph is defined as a quadruple $G = (\Sigma_v,\ \Sigma_e, \gamma_v, \gamma_e)$, where $\Sigma_v = \{v_a \mid a = 1, \ldots, n\}$ is the set of vertices and $\Sigma_e = \{e_{ab} | a, b \in 1, \ldots, n\}$ is the set of edges. Function $\gamma_v : \Sigma_v \to \Delta_v^N$ assigns N attributes $v_a = [v_{a(1)}, \ldots, v_{a(N)}]$ in the domain Δ_v to each node. Similarly, γ_e: $\Sigma_e \to \Delta_e$ assigns M attributes $e_{ab} = [e_{ab(1)}, \ldots, e_{ab(M)}]$ in the domain Δ_e to each edge. The order of graph G is n.

The Star of a node $\mathrm{v_a}$, denotated $\mathrm{S_a}$, on an attributed graph G, is another graph $\mathrm{S_a} = (\Sigma_\mathrm{v}^{\mathrm{S_a}},\ \Sigma_\mathrm{e}^{\mathrm{S_a}}, \gamma_\mathrm{v}^{\mathrm{S_a}}, \gamma_\mathrm{e}^{\mathrm{S_a}})$ composed of the node $\mathrm{v_a}$, the nodes connected to $\mathrm{v_a}$ by an edge and these edges. Formally, $\Sigma_\mathrm{v}^{\mathrm{S_a}} = \{\mathrm{v_a} \cup \mathrm{v_b} | \mathrm{e_{ab}} \in \Sigma_\mathrm{e}\ \}$ and $\Sigma_\mathrm{e}^{\mathrm{S_a}} = \{\mathrm{e_{ab}} | \mathrm{e_{ab}} \in \Sigma_\mathrm{e}\ \}$. Moreover, $\gamma_\mathrm{v}^{\mathrm{S_a}}(\mathrm{v_b}) = \gamma_\mathrm{v}(\mathrm{v_b}),\ \forall \mathrm{v_b} \in \Sigma_\mathrm{v}^{\mathrm{S_a}}$ and $\gamma_\mathrm{e}^{\mathrm{S_a}}(\mathrm{e_{ab}}) = \gamma_\mathrm{e}(\mathrm{e_{ab}}),\ \forall \mathrm{e_{ab}} \in \Sigma_\mathrm{e}^{\mathrm{S_a}}$. The order of star $\mathrm{S_a}$ is n_{N_a}.

Error-tolerant graph matching

Let $\mathrm{G^p} = (\Sigma_v^p,\ \Sigma_e^p, \gamma_v^p, \gamma_e^p)$ and $\mathrm{G^q} = (\Sigma_v^q,\ \Sigma_e^q, \gamma_v^q, \gamma_e^q)$ be two attributed graphs of initial order n and m. To allow maximum flexibility in the matching process, graphs can be extended with null nodes. We refer to null nodes of $\mathrm{G^p}$ and $\mathrm{G^q}$ by $\hat{\Sigma}_v^p \subseteq \Sigma_v^p$ and $\hat{\Sigma}_v^q \subseteq \Sigma_v^q$ respectively. Let F be a set of all possible bijections between the two node sets $\Sigma_\mathrm{v}^\mathrm{p}$ and $\Sigma_\mathrm{v}^\mathrm{q}$. We define the non-existent or null edges as $\hat{\Sigma}_e^p \subseteq \Sigma_e^p$ and $\hat{\Sigma}_e^q \subseteq \Sigma_e^q$. Correspondence $f^{p,q} : \Sigma_v^p \to \Sigma_v^q$ is bijective and assigns one node of $\mathrm{G^p}$ to only one node of $\mathrm{G^q}$.

Graph Edit Distance between graphs (GED)

One of the most widely used methods to evaluate an error-tolerant graph matching is the GED [1,2]. The dissimilarity is defined as the minimum amount of required distortion to transform one graph into the other. To this end, a number of distortions or edit operations, consisting of insertion, deletion and substitution of both nodes and edges are defined. Edit cost functions are introduced to quantitatively evaluate the edit operations. The basic idea is to assign a penalty cost to each edit operation according to the amount of distortion that it introduces in the transformation. Deletion (insertion) operations are transformed to assignments of a non-null node (null node) of the first graph to a null node (non-null node) of the second graph. Substitutions simply indicate node-to-node assignments. Using this transformation, given two graphs G^p and G^q, and a bijection between their nodes, $f^{p,\,q}$, the graph edit cost is:

If we consider $f(v_a^p) = v_i^q$ and $f(v_b^p) = v_j^q$, which forces e_{ab}^p to be mapped to e_{ij}^q, the *EditCost* is,

$$\begin{aligned} EditCost(G^p, G^q, f) = & \sum_{\forall\ v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \quad s.t. \quad v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q} C_{w_v}\left(v_a^p, v_i^q\right) \\ & + \sum_{\forall\ e_{ab}^p \in \Sigma_e^p - \hat{\Sigma}_e^p \quad s.t. \quad e_{ij}^q \in \hat{\Sigma}_e^q - \hat{\Sigma}_e^q} C_{w_e}\left(e_{ab}^p, e_{ij}^q\right) \\ & + \sum_{\forall\ v_a^p \in \hat{\Sigma}_v^p \quad s.t. \quad v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q} K_v \\ & + \sum_{\forall\ e_{ab}^p \in \hat{\Sigma}_e^p \quad s.t. \quad e_{ij}^q \in \Sigma_e^q - \hat{\Sigma}_e^q} K_e \\ & + \sum_{\forall\ v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \quad s.t. \quad v_i^q \in \hat{\Sigma}_v^q} K_v \\ & + \sum_{\forall\ e_{ab}^p \in \Sigma_e^p - \hat{\Sigma}_e^p \quad s.t. \quad e_{ij}^q \in \hat{\Sigma}_e^q} K_e \end{aligned} \tag{1}$$

C_{Wv} is a function that represents the cost of substituting node $\mathrm{v_a^p}$ of $\mathrm{G^p}$ by node $f^{p,q}(v_a^p)$ of $\mathrm{G^q}$. The same think occurs with

Download English Version:

https://daneshyari.com/en/article/11002883

Download Persian Version:

https://daneshyari.com/article/11002883

Daneshyari.com