# SSDDM: Distance Metric for Graph-based Semi-structured Data

**István Soós, Tihamér Levendovszky, Hassan Charaf**
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
H-1111, Goldmann Gy. ter 3.
Budapest, Hungary
email: {istvan.soos, tihamer.levendovszky, hassan.charaf}@aut.bme.hu

*Abstract: Data mining of semi-structured data (data with no exact schema) is an emerging field of interest. Data mining algorithms such as clustering, classification need to have a metric distance for the defined data records. The most natural representation for semi-structured data is the graph-representation with labeled nodes and edges. Graph-edit distance measure can be used for comparing the similarity of two record of data, but semi-structured data can contain several attached attributes, values. It is shown in this paper that a graph distance metric can be extended to semi-structured data, preserving the properties of metric distance. This metric distance (SSDDM) can be used for classification of large data collections and other data mining algorithms.*

*Keywords: data mining, metric distance, semi-structured data*

## 1 Introduction

Data mining, data warehouses, and information extraction algorithms are one of the most important parts of computer automation. It is a non trivial process of discovering previously unknown and potentially useful information from large databases. The data must be converted into a specific, structured format, and structure-aware algorithms are executed for ”mining”. The basic data mining algorithms for these structured data (association rules, classification, clustering) are fairly developed. We use structured data if the storage schema and the information carried by the data is the same. The currently available and the future data records can be written with the same and constant data schema. Relational database management systems are designed and optimized to work with structural data, and decision support systems are planned to work with these relational, structured databases.

By semi-structured data, we mean that although the data may have some structure, the structure is not as rigid, regular, or complete as the structure required by traditional database management systems [1]. If the structure of the data storage does not match the information content defined by the semantics of the data, we can use semi-structured data representation. The most natural representation of semi-structured data is the graph representation with labeled nodes and edges. Each node can hold one or more value, and can be connected with other nodes with directed and labeled edges. The direction of edges indicates the ”hierarchy” of the objects, but it is not a strict hierarchy: it is data and relationship description.

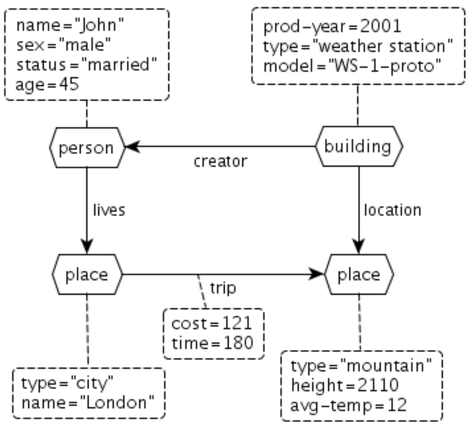

Figure 1. Semi-structured data example

Figure 1 describes a sample semi-structured data scenario fragment. A weather station was created by John, who lives in London. The location of the station is 2110 meters above sea level, in the mountains (other attributes are not displayed on the graph), and the trip from London (from John's home) to the mountain costs €121 and takes 180 minutes. Attributes are attached to vertices (mountain) and edges (trip) as well.
For cluster-based data mining and information extraction, a distance metric is needed. A distance measure $d$ fullfills the properties of a metric:

- $d(A, B) = 0 \iff A = B$
- $d(A, B) = d(B, A)$
- $d(A, B) \leq d(A, C) + d(C, B)$

Graph distance metric [2] is applicable for graph based structures, but not enough for semi-structured data mining algorithms. Applying graph-only metric loses the distance information hidden in attributes eg. the cost of the trip or the age of the creator (John).

## 2 Related Work

The theory, the storage, and query mechanism of semistructured data is elaborated in ([1], [3], [4], [5], [6]). The data mining algorithms for these problems ([7], [8]) are expected to come closer to the structured algorithms in a period of short time. Graphs are not only the data structure layer of semi-structured data, but they are general, useful and powerful data structures in a variety of applications. Computer

vision, molecular research, pattern recognition, software modeling, geographical information systems use graphs as well. These modern computationally and data intensive applications require efficient data storage mechanisms. The storage and query algorithms have to focus on the ”most likely” data retrieval or nearest neighbor search [9].
Clustering and classification algorithms need to have a metric function for measuring the distance between objects. For structured records, these distance functions are evident, because the attributes and the columns of the records can be mapped together, the measurement unit can be normalized. For graph-based structures, the distance function is harder to define. First, a mapping between the vertices need to be calculated for each measured graph pair, and with the mapping information, the attribute-based distance is applicable.
Distance measures are related to graph similarities, graph and subgraph isomorphism. Graph isomorphism ([10], [11], [12]) is a very natural measure of graph similarity, but minor differences could influence the efficiency of the measurement. Errors, distortions, and ”bad-luck” modifications can be substituted by the application of edit distance [13] or histograms of graphs. Using the edit distance, we have to define the deletion, creation and substitution costs of a vertex or edge operation of the graph, but there is no universal cost rate, it depends purely on the application domain what to use. Another way for calculating distance is finding the largest common subgraph of two graphs ([2]). Our work relies on a simple graph distance metric definition [2]. It is shown that the following distance function is metric:

$$d = \frac{|mcs(G_1, G_2)|}{max(|G_1|, |G_2|)} \tag{1}$$

We will show, that it can be extended for a semi-structured domain. The attribute mapping of the vertices will be calculated with the maximal common subgraph mapping, and the distance function will be based on the number of attributes attached.

## 3 Semi-structured Data Graph Distance

At the beginning of this section, a formal definition is shown for semi-structured data. A distance metric will be introduced in two steps: first for weighted graphs, then for semi-structured data defined.

### 3.1 Definitions

A *semi-structured data-graph* (SSD-G) is a 7-tuple $G = (V, E, \mu, \nu, \omega, \phi, \psi)$ where:

- $V$ is a set of finite vertices
- $E \subseteq V \times V$ is the set of directed edges
- $\mu \rightarrow L_V$ is a function assigning labels to the vertices
- $\nu \rightarrow L_E$ is a function assigning labels to the edges

- $\omega : V \rightarrow \{\kappa \rightarrow \upsilon\}$ is a function assigning set of key-value mapping pairs to the vertices
- $\phi : E \rightarrow \{\kappa \rightarrow \upsilon\}$ is a function assigning set of key-value mapping pairs to the edges
- $\psi : \{\kappa \rightarrow \upsilon\}$ is a function assigning set of key-value mapping pairs to the graph

Given a $G = (V, E, \mu, \nu, \omega, \phi, \psi)$ SSD-G, *semi-structured data-subgraph* (SSD-SG) is a 7-tuple $G_S = (V_S, E_S, \mu_S, \nu_S, \omega_S, \phi_S, \psi_S)$ such that:

- $V_S \subseteq V$
- $E_S = E \cap V_S \times V_S$
- $\mu_S : \{(v \rightarrow L_v) \in \mu : \ v \in V_S\}$
- $\nu_S : \{(e \rightarrow L_e) \in \nu : \ e \in E_S\}$
- $\omega_S : V_S \rightarrow \{\kappa' \rightarrow \upsilon'\}$ is a function assigning set of key-value mapping pairs to the vertices and there is no predefined relationship with $\omega$
- $\phi_S : E_S \rightarrow \{\kappa' \rightarrow \upsilon'\}$ is a function assigning set of key-value mapping pairs to the edges and there is no predefined relationship with $\phi$
- $\psi_S : \{\kappa' \rightarrow \upsilon'\}$ is a function assigning set of key-value mapping pairs to the graph and there is no predefined relationship with $\psi$

The SSD-SG is a subgraph of the original graph, without considering the $\omega, \phi, \psi$ informations attached to it. Please note, that the definition we use is very permissive, because our experience shows that a more restrictive definition causes more processing power without any advantage. Restricting $\omega, \phi, \psi$ information, the maximal subgraph matching algorithm will be harder to calculate.

Given a $G_1 = (V_1, E_1, \mu_1, \nu_1, \omega_1, \phi_1, \psi_1)$ and $G_1 = (V_2, E_2, \mu_2, \nu_2, \omega_2, \phi_2, \psi_2)$ SSD-G, *semi-structured data - common subgraph* (SSD-CSG) is a 7-tuple $C = (V_C, E_C, \mu_C, \nu_C, \omega_C, \phi_C, \psi_C)$ such that: $C$ is a SSD-subgraph of $G_1$ and $C$ is a SSD-subgraph of $G_2$.

The SSD-common subgraph is a common subgraph of the original graphs, without the restrictions of $\omega, \phi, \psi$ functions. A common subgraph $C$ of $G_1$, $G_2$ is *maximal*, when there exists no other common subgraph of $G_1$, $G_2$ that has more nodes than $C$. The maximal common subgraph of $G_1$, $G_2$ is denoted with $mcs(G_1, G_2)$.

The definitions are self-descriptive, considering the fact, that all semi-structure definition is about extending the graph definitions with attached attributes. The attached attributes do not conflict with the graph structure or any graph algorithms involved. We refer to *graph isomorphism* and *subgraph isomorphism* as regular graph-related expressions, there will be no distinctions at the domain of semi-structured data.

## 3.2 Graph Distance Metric with Weights

First we show, that the graph distance metric can be extended with node-attached weight attributes. Let $\rho : V \to \{\mathbb{N} \setminus 0\}$ an additional positive weight function for vertices. For a given graph $G = (V, E, \mu, \nu)$ and $\rho_G$ we can construct a new graph $G' = (V', E', \mu', \nu')$ with the following algorithm:

- for each $n \in V$ there is a $\rho_n - 1 \geq 0$
- we create $R = (V_R, E_R, \mu_R, \nu_R)$ such as $|V_R| = r$, $E_R \equiv V_R \times V_R$ and $\mu_R$, $\nu_R$ are universal, unique identifiers for each node and edge. For the case $r = 0$ $R$ will be an empty graph.
- we connect $n \in V$ with $R$, the following way: create a new edge from $n$ to every $n_R \in V_R$ with the label $v_e$, where $v_e$ is a universal unique identifier.

*Universal unique identifiers* can be obtained from a generated pool or sequence, like UUID#1, UUID#2, ... or any other UUID algorithm. For a given $G$ and a single constructed graph $R$, the attachment is denoted by: $R \cdot G$

**Proposition 1.** Given $G_1 = (V_1, E_1, \mu_1, \nu_1)$, $G_2 = (V_2, E_2, \mu_2, \nu_2)$ and $G'$ algorithm above, the maximal common subgraph of the initial graphs will be the same: $mcs(G_1, G_2) = mcs(G'_1, G'_2)$

*Proof:* The $\mu_R$ function is universal and unique, it can not be matched with any other node, the additional $V_R$ nodes do not affect the maximal common subgraph matching, because universal unique identifiers differ from any other identifier by definition.

**Proposition 2.** Given the graph $G_1 = (V_1, E_1, \mu_1, \nu_1)$ and $G_2 = (V_2, E_2, \mu_2, \nu_2)$, their maximal common subgraph $S := mcs(G_1, G_2) = (V_S, E_S, \mu_S, \nu_S)$ and some constructed $R = (V_R, E_R, \mu_R, \nu_R)$ for $n \in S, \rho_S$, and bijective functions for subgraph isomorphism

$$f_1(n_S) = v_{G1} \in V_1 \ \forall n_S \in V_S, f_1^{-1}(v_{G1}) = n_S \in V_S$$

$$f_2(n_S) = v_{G2} \in V_2 \ \forall n_S \in V_S, f_2^{-1}(v_{G2}) = n_S \in V_S$$

If $R$ is attached to $G_1$ and $G_2$ at the nodes $v_{n,1} = f_1(n)$ $(G'_1)$ and $v_{n,2} = f_2(n)$ $(G'_2)$, the vertex count for the maximal common subgraph is: $|mcs(G'_1, G'_2)| = |mcs(G_1, G_2)| + |R|$.

*Proof:* It follows from the definition of maximal common subgraph: $R = mcs(R, R)$, $mcs(G_1 \cdot R, G_2 \cdot R) = mcs(G_1, G_2) \cdot R$, because $R$ is distinct graph from both $G_1$ and $G_2$.

**Proposition 3.** Given $G_1, G_2, S, n, f_1, f_2, v_{n,1}, v_{n,2}$ as Proposition 2, the constructed graph $R_1 = (V_{R,1}, E_{R,1}, \mu_{R,1}, \nu_{R,1})$ for $G_1$, $R_2 = (V_{R,2}, E_{R,2}, \mu_{R,2}, \nu_{R,2})$ for

$G_2$ and their maximal common subgraph $R = mcs(R_1, R_2)$ $|R| > 0$. After attaching $R_1$ and $R_2$ to $G_1$ and $G_2$, the maximal common subgraph is the following: $mcs(G_1 \cdot R_1, G_2 \cdot R_2) = mcs(G_1, G_2) \cdot mcs(R_1, R_2)$ and $|mcs(G_1 \cdot R_1, G_2 \cdot R_2)| = |S| + |R|$

*Proof:* It follows from the definition of maximal common subgraph, because $R_1$ and $R_2$ are distinct graphs from both $G_1$ and $G_2$, they are attached to the identical (mapped) vertex, and they only extend the graph matching with their maximal common subgraph.

**Proposition 4.** Attaching graph $R$ to either of the graph $G_1$ or $G_2$ affects the maximum common subgraph in one of the ways mentioned above (Proposition 1-3). Given the the weight function $\rho_G$ and its bidirectional mapping with the graph $R$, a new graph distance metric can be defined with weight functions for nodes, based on the graph metric defined in [2]. Given graphs $G_1 = (V_1, E_1, \mu_1, \nu_1, \rho_1)$ and $G_2 = (V_2, E_2, \mu_2, \nu_2, \rho_2)$, their maximal common subgraph is $S = (V_S, E_S, \mu_S, \nu_S, \rho_S) := mcs(G_1, G_2)$, where the weight function $\rho_S$ has the following property:

$$\rho_S \leq min(\rho_1(f_1(n)), \rho_2(f_2(n))) \;\; \forall n \in V_S \tag{2}$$

The graph distance metric with weights is defined the following:

$$d(G_1, G_2) = 1 - \frac{\sum\limits_{n \in V_S} \rho_S(n)}{max\left(\sum\limits_{v \in V_1} \rho_1(v), \sum\limits_{u \in V_2} \rho_2(u)\right)} \tag{3}$$

*Proof:* The sum of weights can be originated in graph construction and the graph metric in [2] as follows:
We create the unique extensions for graphs as the algorithm in be beginning of this chapter: $G_1'$ for $G_1$, $G_2'$ for $G_2$, $S'$ for $S$ with the special property for each vertex: $\forall n \in V_S$ $R_n = mcs(R_{f1(n)}, R_{f2(n)})$ and $|R| = r = \rho_S(n) - 1$. We can create such an attached graph, because $\rho_S(n) \leq min(\rho_1(f_1(n)), \rho_2(f_2(n)))$ $\forall n \in V_S$. Obviously, $S'$ is the maximal common subgraph of the two new graphs $S' = mcs(G_1', G_2')$.

It is an unambiguous mapping from the weight of the graphs and subgraph to the special constructed graphs and subgraph. But for non-weighted graphs, we have a graph metric (1) [2]. The metric for the constructed graphs is equivalent to the distance function for weighted graphs.

## 3.3 Semi-structured Distance Metric

In this section is is shown that the semi-structured data graph can be expressed as the discrete weight graph above, then the generalization of the continuous range is discussed.

For a given $G = (V, E, \mu, \nu, \omega, \phi, \psi)$ SSD-G we define $\rho$ the following: $\rho' = |\psi|$ for the attached graph properties, and $\rho(n \in V) = 1 + |\omega_n| + |\phi_{n \to V}|$ for the vertices and edges.

**Proposition 5.** For a given $G_1$ and $G_2$ SSD-G and their maximal common subgraph $S$ with the properties $\omega_S = \omega_1 \bigcap \omega_2$ $\phi_S = \phi_1 \bigcap \phi_2$ $\psi_S = \psi_1 \bigcap \psi_2$ the following distance is a metric:

$$d(G_1, G_2) = 1 - \frac{\rho'_S + \sum\limits_{n \in V_S} \rho_S(n)}{max\left(\rho'_1 + \sum\limits_{v \in V_1} \rho_1(v), \rho'_2 + \sum\limits_{u \in V_2} \rho_2(u)\right)} \tag{4}$$

*Proof:* For $\rho' = |\psi|$ we can define a new, attached and distinct $R$ for any graph. It is not attached to vertices as the attachments before, it is connected separately, and is the property of the graph only. We have new graphs and subgraph, and Proposition 4 is applicable, because of (2).

**Proposition 6.** $\rho : V \to \mathbb{Q}^+$ is applicable.

*Proof:* Every $\rho_1, \rho_2$ and $\rho_S$ value has a common denominator: $d$. If all $\rho$ is multiplied by $d$, Proposition 5 is applicable, because $d \cdot \rho_i \in \mathbb{N}^+$.

**Proposition 7.** For a given $G_1$ and $G_2$ SSD-G and their maximal common subgraph $S$ with the attribute distance function $\rho_S(\{\kappa \to \upsilon\}) = \sum \alpha_\kappa(\upsilon_1, \upsilon_2)$ where $0 \leq \alpha_\kappa \leq 1$, $\alpha_\kappa \in \mathbb{Q}^+$ and $\alpha_\kappa(\upsilon_1, \upsilon_2) = 0$ if $\upsilon_1$ or $\upsilon_2$ is not defined, the following distance is metric *(SSDDM)*:

$$d(G_1, G_2) = 1 - \frac{\rho'_S + \sum\limits_{n \in V_S} \rho_S(n)}{max\left(\rho'_1 + \sum\limits_{v \in V_1} \rho_1(v), \rho'_2 + \sum\limits_{u \in V_2} \rho_2(u)\right)} \tag{5}$$

*Proof:* It follows from Proposition 5 and 6.

## 4 SSDDM Algorithm

We introduce a simple pseudo-code description of the distance algorithm. The notation and the syntax is simplified. Furthermore, we assume familiarity with an object-oriented language.

```
function double SSDDM (Graph G1, Graph G2) {
   // calculating MCS
   Mcs_Data mcs=maxCommonSubgraph(G1, G2);
   // subgraph
   Graph S = mcs.getSubGraph();
   // mapping informations (f1, f2)
   Graphmapping mapS_G1 = mcs.getMap1();
   Graphmapping mapS_G2 = mcs.getMap2();

   // setting counters
   double rho_G1 = calculateFullRho(G1);
   double rho_G2 = calculateFullRho(G2);
   double rho_S = S.numberOfVertices();

   // calculating rho of subgraph
   foreach (Vertex v_S in S) {
      // get mapped vertex of graphs
      Vertex v_G1 = mapS_G1.get(v_S);
      Vertex v_G2 = mapS_G2.get(v_S);

      // get omega and phi attributes
      Map om_G1 = v_G1.getOmega();
      Map om_G2 = v_G2.getOmega();
      Map phi_G1 = v_G1.getPhi();
      Map phi_G2 = v_G2.getPhi();

      // intersect attributes
      rho_S += intersect(om_G1, om_G2);
      rho_S += intersect(phi_G1, phi_G2);
   }

   // intersect psi attributes
   rho_S += intersect(G1.getPsi(), G2.getPsi());
   // calculating distance
   return 1-( rho_S / max(rho_G1, rho_G2));
}

// intersection is for calculating the
// common parts of the attribute maps
function double intersect(Map m1, Map m2) {
   // result is the distance of the
   // map parameters
   double result = 0;
   // we have to iterate on one
   // of the maps
   foreach (Key k in m1) {
      Value v1 = m1.get(k);
      Value v2 = m2.get(k);
      // invoking alpha function
      // based on key and values
      result += alpha(k, v1, v2);
   }
   return result;
}

// calculating rho for a full graph
// is counting the number of nodes
// and the size of omega, phi, psi
// attributes
function double calculateFullRho(Graph g) {
   double result = 0;
   // adding number of vertices
   result += g.numberOfVertices();

   // iterating vertices for
   // omega and phi
   foreach (Vertex v in g) {
      result += v.getOmega().size();
      result += v.getPhi().size();
   }
   // adding psi
   result += g.getPsi().size();
   return result;
}
```

## 5 SSDDM Calculation Example

The distance function is explained on the following example. The $G_1$ graph will be the graph on Figure 1 ($v_1$ = person: John, $v_2$ = place: London, $v_3$ = place: mountain, $v_4$ = building: weather station) and $G_2$ will be the graph on Figure 2 ($u_1$ = person: Mary, $u_2$ = place: New York, $u_3$ = organization: MegaCorp).

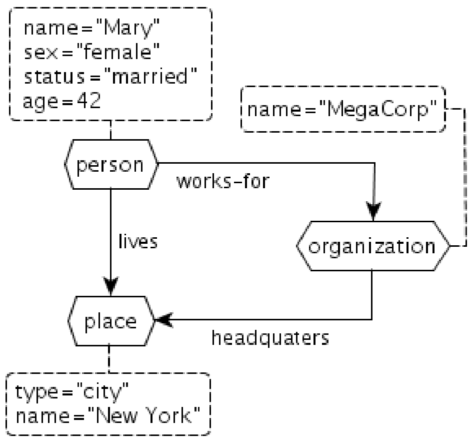

Figure 2. Other SSD-graph example

The maximal common subgraph calculation gives the result of two vertices we present in Figure 3.

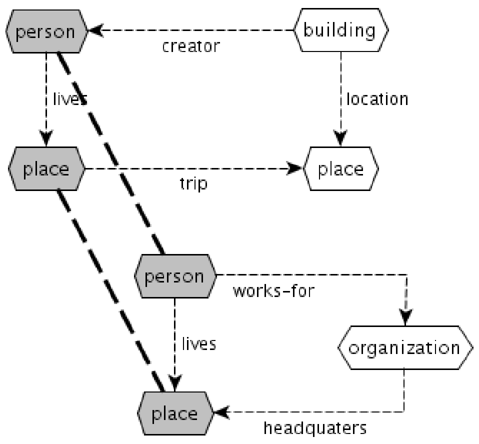

Figure 3. Maximal common subgraph calculation

Based on the graph distance metric (1) the distance measure is the following:

$$d(G_1, G_2) = 1 - \frac{2}{max(3,4)} = 0.5$$

If we consider the attached attributes of the graph (5), the distance measure will be the following:

$$\begin{aligned}
\rho'_S &= 0,\ \rho'_1 = 0,\ \rho'_2 = 0 \\
\sum_{v \in V_1} \rho_1(v) &= 6 + 6 + 4 + 6 = 22 \\
\sum_{u \in V_2} \rho_2(u) &= 7 + 3 + 3 = 13 \\
\sum_{n \in V_S} \rho_S(n) &= (2 + \alpha(v_1, u_1)) + (1 + \alpha(v_2, u_2)) \\
\alpha(v_1, u_1) &= 0 + 0 + 1 + \frac{60 - |45 - 42|}{60} = 1.95 \\
\alpha(v_2, u_2) &= 1 + 0 = 1 \\
\sum_{n \in V_S} \rho_S(n) &= (2 + 1.95) + (1 + 1) = 5.95
\end{aligned}$$

$$d(G_1, G_2) = 1 - \frac{5.95}{max(22,13)} = 0.7295$$

In this example the distance measure that we have constructed indicates less similarity (= more distance) than the distance measure calculating only with graph vertices. Based on the semantics of the data, it is fairly correct. The common subgraph shows that the common part of the two graphs is the following: we have two married, almost the same age (42-45) people, who are living in a city. Everything else is different: their names, sex, living place; the first one built a weather station, the second is working at a company - the information are not related.

The more details we consider in the similarity measure, the more precise it is. Increasing the common attributes in the maximal common subgraph decreases the distance. An other example can be constructed where the graph distastance is larger than the SSD-distance.

## Conclusion

We have defined a new distance metric (SSDDM) for semi-structured data. A distance metric is widely used for data mining algorithms like clustering and classification, but these subjects are less elaborated at the semi-structured domain. We have examined the following application domains for the distance metric defined:

Plagisation detection for large, object-oritented programming languages is hard. A student can copy homework and refactor it many ways, and for projects like J2EE, it is almost impossible to detect these efforts. But these architectures have some significant spots: service entry points, delegation patterns, which help us detect similarities. The source code can be compiled to a syntax tree, and based on variable and function references, it can be extended to a graph. After we have the program-reference graph, the significant spots, the entry points and delegations can be extended with additional attributes, and variables can be marked with usage patterns. The defined distance metric enables us to concentrate on structural similarities and code execution paths instead of tree-refactoring methods.

Automatic language processing task (e.g. translation, information extraction) are working with grammar-analized texts. The tree-building grammars are constrained, and less effective for languages like Hungarian or Korean, cross-reference and ontology extensions are required. These structures extend the plain grammar tree to a more detailed analysis of the text with attached attributes. Therefore the data structure can be considered as semi-structured data. Translation engines lack of learning capabilities, because language patterns are not restrictive enough. The distance metric function above can help to calculate similarities between sentences and segments, and increase the precision of translations.

Other information extraction task is searching for similarities in an encyclopedia. The most common, community-edited encyclopedia on the Internet is Wikipedia. We have transformed the Hungarian Wikipedia articles to semi-structured graphs based on the titles, links, positions of the text, and measured the distances of the pages based on the $n$-step depth level graph of the pages.

Scenarios with large problem-space (like the board game Go) can be reduced to a less schema-aware, semi-structured graph. Extraction of rules (e.g. board evaluation) can be simplified with the distance metric above, because the search for similar

problem configuration (e.g. board state) can be found in a less detailed environment. The only limitation of the distance metric is the complexity of the maximal common subgraph calculation.
The advantage of the metric is the simplicity of the calculation: after finding the maximal common subgraph, the weight of the attached attributes can be calculated in a straightforward way. Comparing to a standard graph-metric, attached attributes are taken in account, in our novel approach more details are calculated for the distance, and the precision increases.

## References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener, "The Lorel query language for semistructured data," *International Journal on Digital Libraries*, Vol. 1, No. 1, pp. 68–88, 1997

[2] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recognition Letters*, Vol. 19, 1998

[3] S. Abiteboul, "Querying semi-structured data," in *ICDT*, pp. 1–18, 1997

[4] S. Abiteboul, P. Buneman, and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000

[5] S. Abiteboul and P. Kanellakis, "Object identity as a query language primitive," *ACM SIGMOD*, May 1989

[6] P. Buneman, M. F. Fernandez, and D. Suciu, "UnQL: a query language and algebra for semistructured data based on structural recursion," *VLDB Journal: Very Large Data Bases*, Vol. 9, No. 1, pp. 76–110, 2000

[7] K. Abe, S. Kawasoe, T. Asai, H. Arimura, and S. Arikawa, "Optimized substructure discovery for semi-structured data," *PKDD02*, 2002

[8] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," *ICDM*, 2002

[9] P. A. and M. Y., *Nearest Neighbor Search: A Database Perspective*. Series in Computer Science, 2005

[10] J. Ullman, "An algorithm for subgraph isomorphism," *Journal of the ACM*, 1976

[11] R. Read and D. Corneil, "The graph isomorphism disease," *Journal of Graph Theory*, 1977

[12] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "An improved algorithm for matching large graphs," *3rd IAPR-TC15 Workshop on Graph-based Representation*, 2001

[13] H. Bunke, "On a relation between graph edit distance and maximum common subgraph," *Pattern Recognition Letters*, 1997