A LRAAM-based Partial Order Function for Ontology Matching in the Context of Service Discovery

Hendrik Ludolph[1], Peter Kropf[1] and Gilbert Babin[2]
[1]*Institute of Computer Science, University of Neuchâtel, 2015 Neuchâtel, Switzerland*
[2]*Information Technologies, HEC Montréal, 3000, ch. Côte-Ste-Catherine, Montréal (QC) H3T 2A7, Canada*

Keywords: Ontology Matching, Neural Network, Service, Integration.

Abstract: The demand for *Software as a Service* is heavily increasing in the era of *Cloud*. With this demand comes a proliferation of third-party service offerings to fulfill it. It thus becomes crucial for organizations to find and select the right services to be integrated into their existing tool landscapes. Ideally, this is done automatically and continuously. The objective is to always provide the best possible support to changing business needs. In this paper, we explore an artificial neural network implementation, an LRAAM, as the specific oracle to control the selection process. We implemented a proof of concept and conducted experiments to explore the validity of the approach. We show that our implementation of the LRAAM performs correctly under specific parameters. We also identify limitations in using LRAAM in this context.

1 INTRODUCTION

Today, more than ever, Information Technology (IT), and more specifically Information Systems (IS) are necessary for an organization to succeed. An IS can be defined as a specific assembly of applications to support distinctive enterprise needs (Izza, 2009). As an organization evolves, so does the IS supporting it. Many factors put pressure on the organization, which in reaction will evolve. The sources of this pressure include, but are not limited to, competition, internal and external politics, organizational evolution, technical progress, and cost containment. The changes induced by these pressure sources lead to a reconfiguration of the organization, which in turns leads to a reconfiguration of the IS supporting it. We anticipate that these changes will tend to occur more frequently and more rapidly. The high frequency of technological innovations brought to the market illustrates this tendency. This is further enhanced by the presence of cloud-based solutions, i.e. SaaS. As a consequence of this trend, we must find ways to increase our ability to evolve the organization's IS as often and as quickly as it is required to maintain the stability of the organization. In a perfect world, the IS itself would "know" when change is required and would "adapt" itself to better fit the needs of the organization. This might only occur if the IS has some understanding of what the organization's requirements were and how they evolved. To adapt adequately, the IS must be able to identify what are the alternatives from which to select the most appropriate response to the change. The research presented in this paper is set in that vision of autonomic IS adaptation in the context of *service oriented architecture* (SOA).

The basic idea of SOA is to modularize and wrap applications behind formally described access points or interfaces, e.g., Application Programming Interfaces (APIs) which follow more or less rigorous protocols (e.g., SOAP, REST, JSON (Erl et al., 2014)) and are accessible over the network. Through the APIs, applications' functionalities can be automatically discovered (e.g., using UDDI registries; see (Kale, 2014)) and consumed as a *service*. Services can represent anything from simple service requests to complicated business processes (Lehaney et al., 2011). They can participate in many different IS (Kale, 2014). With SOA, eventually, the objective is to lower integration hurdles and increase reusability of applications. It empowers organizations to assemble complex IS with unprecedented flexibility and sophistication as business requirements shift over time (Erl, 2004).

The SOA principle is applicable beyond organizational limits. Specialized service providers, such as SalesForce, ServiceNow, Akamai, etc. emerged to extend on-premise SOA to off-premise cloud-based services. They commoditize and commercialize ser-

Ludolph, H., Kropf, P. and Babin, G.
A LRAAM-based Partial Order Function for Ontology Matching in the Context of Service Discovery.
DOI: 10.5220/0006294904210431
In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pages 393-403
ISBN: 978-989-758-243-1

vices, such as Customer Relationship Management, IT Service Management, Performance & Availability. Other organizations request, contract, and integrate these services into their internal IS instead of setting up the functionality by themselves. This way, no expensive technical know-how for the service is needed. If later the service is no longer useful, the contract is cancelled. The service, technically and commercially, disappears from the organizational scope. This flexibility appeals to more and more organizations these days to improve their IS (Cisco, 2014). Some authors even claim that it becomes mandatory for keeping a competitive advantage (Fensel and Bussler, 2002; Hussain and Mastan, 2014). The commercial success of some service providers, such as SalesForce, acts as an incentive for new firms to enter the SaaS market (see (Frank and Cartwright, 2013, Chapter 11) – *long-term zero-profit equilibrium*) and offer similar (sometimes identical) services. This leads to a proliferating number of *similar* cloud-based services to choose from (Bughin and Chui, 2010).

In this context, the organization's IS is an assembly of services (in-house or cloud-based). It provides a more flexible and easier to adapt solution to support all requirements of the organization. Hence, whenever a change in the organization's requirements occurs, the set of all services available is searched to select those which may best support the changed requirements, and to remove the obsolete/inappropriate services from the IS scope, and finally integrate the newly selected services.

Current industrial integration techniques, such as traditional middleware (e.g., remote procedure call mechanisms, data-oriented, component-oriented, message-oriented, application servers), EAI tools (e.g., MS Biztalk, Tibco), BPM (e.g., BPEL, BPMN), or SOA (Oracle Service Bus) do support service integration. However, these approaches do not lend themselves to autonomic IS adaptation. For many authors (Izza, 2009; Fensel et al., 2011; Hoang et al., 2014; Hoang and Le, 2009), these integration techniques are agnostic to the most crucial aspect of autonomic IS adaptation, that is, understanding the semantics of services. Indeed, these techniques merely regulate information and focus on meta-data exchange. They are syntactic in nature. The same authors suggest using *Semantic Web*-based approaches, which enable machines to "understand" the meaning, i.e., the semantics of services. In our view, semantic understanding is required both to determine changes in requirements, comparing the old and the new requirements, and how to resolve the differences, identifying and selecting appropriate services.

The work presented in this paper focuses on one of the tasks that must be performed by the autonomic IS system: service selection and composition (SSC). In our view, SSC should (1) automatically select the "best" service available; (2) automatically integrate the selected service to the organization's IS; and (3) do this continuously as part of the autonomic IS environment. All this is based on the premise that we can automatically determine how well a service supports requirements, and by extension, that we can rank services by their level of support. It is clear that the selection process goes beyond simple discovery as it represents a degree of intelligence, namely identification and analysis towards synthesis of possible actions (Zdravković et al., 2014). We further limit the scope in this paper to the selection process itself. Indeed, once selection is performed, existing SOA techniques can be used to facilitate/automate the actual integration process.

Specifically, the paper presents exploratory results on a novel service selection approach which uses ontological description of services. That is, we assume that both organization's requirements and service offerings are described and represented using an ontological notation, such as OWL (Web Ontology Language), in addition to the usual descriptions used for SOA (e.g., UDDI, SOAP, etc.), which are by nature syntactic, as they describe the APIs and the data structure, but do not provide any information about the purpose of the service, at least not in a form that can be processed by a computer. In (Ludolph et al., 2011), the authors present a global service selection algorithm. The algorithm assumes the existence of a service matching function. In this paper, we focus on the definition of the service matching function, which lacked in (Ludolph et al., 2011), using an ontology matching approach. The main issue addressed in the paper, therefore, pertains to the specification and analysis of a partial order function used to rank services from most appropriate to least appropriate based on the similarity of their ontological description.

We first start with a brief analysis of the use of ontology matching approaches in the context of service composition (Sec. 2). This sets the context in which the partial order function is required. We then explore different alternative approaches to define such a partial order function. Established symbol-driven (Sec. 2.2) as opposed to neural network matching techniques (Sec. 2.3) are discussed. From the latter category, in Section 3, the LRAAM, a specific type of artificial neural network, is further investigated. In Sections 4 and 5, we describe experiments using an LRAAM-based partial order function to perform ontology matching. The paper is concluded in Section 6 with a critical discussion of the results, the approach's

limitations and a outlook on future research.

2 THE MATCHING PROBLEM

Following (Fensel et al., 2011) and (Born et al., 2007), the semantic descriptions of services is necessary in order to establish and warrant interoperability that does not require a human to manually effect certain integrations that will rapidly become obsolete, or non-reusable in a dynamically evolving environment. The approach described in this paper thus focuses on a semantics-based approach towards more intelligent SSC. We assume that business activities (e.g. a task in BPMN terminology), and how they are related, and that independent, competing services are described using semantic descriptions. In our context, we focus on what a *service* provides as opposed to on how to access it technically (e.g., its UDDI description).

A common approach to supply semantic decriptions is the use of ontologies. An ontology is a formal representation of some knowledge domain. It contains relevant entities and their relations. It is based on formal semantics, i.e., logic, allowing for machine reasoning (see (Antoniou and van Harmelen, 2008; Antoniou and van Harmelen, 2009) for a detailed introduction). Figure 1 illustrates such an ontological description, where we can identify five distinct instances of *Activity*. A similar ontological description is assumed for services.

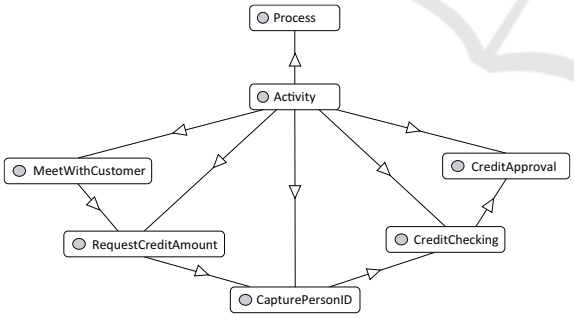

Figure 1: Simple ontology for related business activities.

Provided that the organizational requirements descriptions are available in a standardized form (e.g., OWL-based ontologies), we assume that the description of a business activity will be more or less similar to a corresponding service. Thefore, "similarity" could be used as a selection criterion. It follows that the most similar service for a certain business activity will be the one selected. Eventually, once all services required are selected, service composition can be accomplished by adequate attribute mapping. These mappings could be automatically constructed or adapted through syntactic or semantic matching techniques (Euzenat and Shvaiko, 2013).

2.1 Finding a Good Oracle for Ontology Matching

Automatically establishing service–requirement mappings on a large scale remains a challenge (Diallo, 2014; Otero-Cerdeira et al., 2015). To this we add other challenges: (1) the number of distinct ontologies to evaluate in a continuous manner, considering the evolving organizational context described above, (2) the *efficiency* in terms of search space and time consumption, (3) the *effectiveness* in terms of correct and complete identification of semantic correspondences (Rahm, 2011), (4) the potential use of possibly fragmented background knowledge, and (5) the user involvement (Shvaiko and Euzenat, 2013).

Matching is the fundamental operation to identify similarities (Rahm and Bernstein, 2001) between two ontologies. It takes two ontologies as input and produces a mapping between semantically similar entities. Following the approach described in (Ludolph et al., 2011), we start from an ontological description of a business process (such as in Fig. 1) to identify all activities (using a "is-a" relationship). Then, we look at all valid pairs of consecutive activities $\{aa'\}$ in the business process (using a "preceeded by" relationship). In this context, activity a is a predecessor of a' in the process. We also consider their contextual use within the business process. We seek to identify a pair of services $\{ss'\}$ that supports the most adequately the sequence of activities $\{aa'\}$. Services are also identified using a "is-a" relationship in the service ontologies. A generic ontology fragment, called a sequence ontology, is used to represent the precedence relationship (Ludolph et al., 2011). The sequence ontology has two placeholders, one for the predecessor activity (service), and one for the successor activity (service). Using the sequence ontology, we construct the set $R = \{r_{aa'}\}$ of reference ontologies by replacing both placeholders by a and a', respectively. In the same way, we construct the set $C = \{c_{ss'}\}$ of compound ontologies representing the composition of services s and s', such that $s \neq s'$.

Reference and compound ontologies are compared against one another to evaluate a matrix $D = [d_{rc}]$, where d_{rc} is the ontological distance between reference ontology r and compound ontology c. In this context, a distance $d_{rc} = 0$ would yield identical ontologies r and c, while the value of d_{rc} would increase as the dissimilarity between r and c increases. Using the ontological distance, the general matching algorithm (Ludolph et al., 2011) then tries

to find an optimal solution which identifies the best matches among R and C and which minimizes (1) integration costs, (2) costs of on-premise applications packages/add-on's providing services bundles, and (3) costs of off-premise, cloud-based services. The optimal solution must fulfill the following constraints: (1) each r must be matched with exactly one c, (2) at most one c is matched with an r, and (3) all service sequences must be coherent with the business activity sequences.

Under these conditions, a perfect set of services, that is, one for which $d_{rc} = 0, \forall r \in R, \forall c \in C$, would return an optimal, integrated sequence of services to support a predefined business process. Integration costs would be negligible.

2.2 Symbol-based Methods to Match Ontologies

The real challenge not addressed in (Ludolph et al., 2011) is in defining explicitly a distance function d_{rc}. Indeed, the authors hypothesize that such a function exists. In general, distance can be defined and measured in different ways. An example is the *Hamming distance*. To obtain it, one counts the minimum number of letter substitutions required to transform one string into another string – the fewer the substitutions, the more similar the strings, the smaller the distance. For example, for two strings a = 'ibm' and b = 'hal', $d_{Ham}^{a,b} = 3$. Using a symbol-based approach, determining an ontological distance is somewhat equivalent to calculating the Hamming distance.

Ontology matching designates the process of finding semantic similarities. Following (Kotis et al., 2006), the matching of two ontologies o and o' can be defined as a morphism from o to o'. One approach to determine a distance would therefore be to determine how many steps are optimally required to morph from ontology o to ontology o'.

The associated matching task is to find an alignment A between o and o' (Euzenat and Shvaiko, 2007). An alignment is a set of correspondences, which in turn is a 4-tuple $\langle id, e_o, e_{o'}, r \rangle$, with id as correspondence identifier, e_o and $e_{o'}$ as entities (e.g., classes, properties) of the compared ontologies, and $r \in \{\leqslant, =, \geqslant, \perp\}$[1] the identified relation (Atencia et al., 2011). Various techniques are used to find correspondences. Table 1 presents a classification into syntactic and (formal) semantic matching techniques.

These techniques focus on individual elements or whole structures. They may analyze frequency distribution or specific languages' morphologies. They introduce external data repositories. Eventually, they all work on discrete arbitrary objects, that is, symbols. They rely on the exactness of the analyzed representations to draw appropriate conclusions.

This, however, is at odds with the fact that the representation of requirements is by nature imprecise. These approaches are therefore insufficient to properly determine the distance.

2.3 LRAAM as Matching Function

To overcome the shortcomings of symbol-based matching techniques in the context of imprecise requirements, we combine them with a non-deterministic approach, namely an artificial neural network (ANN (Hinton et al., 1986)) implemented as LRAAM (Labelled Recursive Auto-Associative Memory (Sperduti, 1993)). Typical symbols in a symbolic model are letter strings. They may be placed into structured relationships with other symbols, e.g., subsumption. However, they do not possess internal structure on their own (Blank et al., 1992). In contrast, ANNs incorporate distributed representations (Chan, 2003), also called subsymbolic representations. These representations might evolve into different patterns, but nevertheless, still behave in a way related to the original pattern. Structure is thus inherent to the representation, also called microsemantic (see Tab. 2). Our view is that in a highly dynamic environment it is not about finding an exact, but rather a most similar service to support an activity. To support this view, a fine-grained (continuous) distributed pattern as opposed to a coarse-grained (discrete) symbolic pattern is evaluated. ANN's properties do not arise from the nodes' individual functionality but from the collective effects resulting from the nodes' interconnections. It amounts to the development of distributed internal representations of the input information (Blank et al., 1992).

An ANN has the ability to derive patterns from complex or changing input data. In this context, it is used to evaluate similarity of ontologies, such as r and c, which may be too difficult to be noticed by either humans or symbol-based computing techniques (Li et al., 2012). Specifically, the ANN is able to learn all of r's, respectively c's, ontological entities and relations at the same time as distributed representations. It allows for changing the matching approach from a coarse-grained $d_{rc} \in \{0, \cdots, u\}$ with $u \in \mathbb{N}$ to a fine-grained $d_{rc} \in [0, u]$ with $u \in \mathbb{R}$ (see also Sect. 3).

The LRAAM is a particular implementation of an ANN. Its most interesting feature is the potential to

[1] reads: is less general than, equal, is more general than, disjoint from.

Table 1: Classification of matching techniques (adapted from (Euzenat and Shvaiko, 2013)).

	Syntactic	Semantic
Element-level	**Informal resource-based** directories, annotated resources **String-based** name similarity, description similarity, global namespace **Language-based** tokenisation, lemmatisation, morphology, elimination, lexicons, thesauri **Constraint-based** type similarity, key properties	**Formal resource-based** upper-level ontologies, domain-specific ontologies, linked data
Structure-level	**Taxonomy-based** taxonomy structure **Graph-based** graph homomorphism, path, children, leaves **Instance-based** data analysis, statistics	**Model-based** SAT solvers, DL reasoners

Table 2: Symbolic vs. subsymbolic paradigm (Blank et al., 1992).

	Subsymbolic	**Symbolic**
Representation	distributed continuous emergent use affects form	atomic discrete static arbitrary
Composition	superimposed context-sensitive	concatenated systematic
Functionality	micro-semantic holistic	macro-semantic atomic

encode (and decode for that matter) labeled directed graphs of arbitrary size (de Gerlachey et al., 1994; Sperduti, 1993). The resulting patterns are sensitive to the graph they represent. Following (Ellingsen, 1997), these patterns can be exploited for similarity analysis. They are thus used to calculate the distance matrix $D = [d_{rc}]$. The general architecture of an LRAAM is shown in Figure 2. It is a supervised 3-layer feedforward network trained by backpropagation. The dashed arrows indicate that this auto-associative architecture[2] must be used recursively (Pollack, 1990). Certain node values from the hidden and output layer are fed back to the input data until the network has reached a steady state, that is, until the activation thresholds remain stable even when feeding new inputs to the network.

The training of the LRAAM is achieved through backpropagation so it learns an identity function $F : \mathbf{x} \rightarrow \mathbf{x}'$, where $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$. A node vector is compressed by using the function $F_c : \mathbf{x} \rightarrow \mathbf{z}$. Then, the compressed representation is reconstructed using the function $F_r : \mathbf{z} \rightarrow \mathbf{x}$. The node vector $\mathbf{x}'$ is thus an approximated output equal to $\mathbf{x}$. The network is trained by presenting the input vectors repeatedly, one vector at the time.

[2]The "auto-association" is a consequence of the equality of input and output layers.

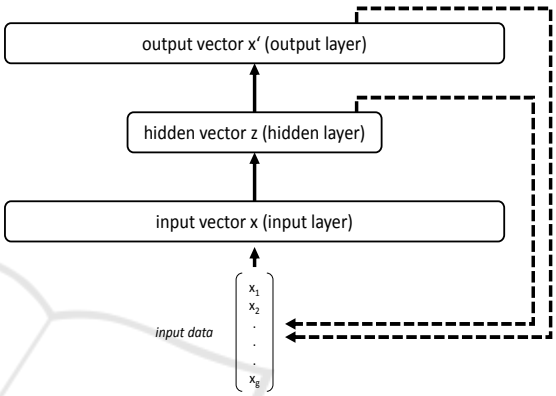

Figure 2: LRAAMs network architecture.

3 INTEGRATING AN LRAAM MATCHING FUNCTION IN THE ONTOLOGY MATCHING ALGORITHM

For each $r \in R$ and $c \in C$, we construct a directed acyclic graph $G = (V, E)$, such that vertices (V) correspond to concepts from the ontology and edges (E) correspond to relationships between these concepts (Eder and Wiggisser, 2007). In other words, G is a conceptual graph. The vector $\mathbf{z} \in \mathbb{R}^m$ serves as comparison pattern and $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\hat{g}}, \ldots, \mathbf{z}_g]^T$ as a collection of comparison patterns, with g the number of vertices in G. Vertices 1 through $\hat{g}$ correspond to the nodes from the generic sequence ontology from which both r and c were constructed. We will note $\mathbf{Z}_{\hat{g}}$ the vector composed of values $[\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\hat{g}}]^T$.

By construction of the LRAAM, we know that each $\mathbf{z}$ in $\mathbf{Z}^r_{\hat{g}}$ and $\mathbf{Z}^c_{\hat{g}}$ contains (micro-semantic) information about the complete collections $\mathbf{Z}_r$ and $\mathbf{Z}_c$, notwithstanding the number of vertices in r or c. Therefore, d_{rc} can be calculated as the Euclidian distance between $\mathbf{Z}^r_{\hat{g}}$ and $\mathbf{Z}^c_{\hat{g}}$. The ontological distance

between r and c is thus defined as:

$$d_{rc} = \sqrt{\sum_{i=1}^{\hat{g}} (\mathbf{z}_i^r - \mathbf{z}_i^c)^2}$$

To construct the LRAAM, we proceed as follows (Fig. 3). Each vertex within G serves as a single input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$. By extension, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_g]^T$, the collection of vertex vectors for a specific r, respectively c, comprises the complete input data to the network.

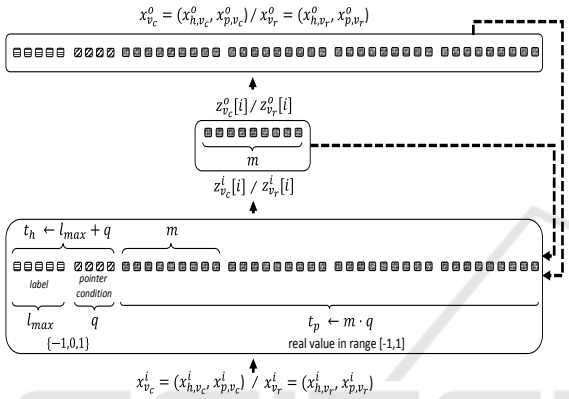

Figure 3: Experimental LRAAM implementation.

To obtain $\mathbf{z}$ of a vertex, part of the input (output) vector is allocated to represent the vertex label and the existence of pointers p (edges[3]) to connected vertices. There are q pointer slots reserved, where $q = max\{degree(v)\}$, with $v \in V$. The input vector is composed of $t_h + q \cdot m \rightarrow \mathbb{R}^n$ elements, where t_h is the number of elements used to represent vertex information (label plus pointer existence), and m is the number of elements used to represent pointer values. If a vertex has less than q pointers, a *nil* pointer is used, to which a random value is initially assigned (Ellingsen, 1997). The hidden representation $\mathbf{z}$ of a specific vertex is understood as the pointer for that vertex. As part of other input vectors, it will thus be used as pointer and iteratively fed to the network. Eventually, we obtain the collection $\mathbf{Z}$ of fixed-sized vectors which represents all concepts/relationships from the ontology. For each reference ontology r, we can thus determine the most similar compound ontology $\overline{c} \in C$, such that $d(\mathbf{Z}_{\hat{g}}^r, \mathbf{Z}_{\hat{g}}^{\overline{c}}), \forall r \in R, \forall c \in C : \mathbf{Z}_{\hat{g}}^r \times \mathbf{Z}_{\hat{g}}^c \rightarrow \mathbb{R}^m$ is minimized.

In a final step, $\overline{c}$ is selected for effective integration. Again, the assumption is that the selected service sequence $\overline{ss'}$ is semantically closer to the sequence of activities aa' and thus better suited to support it.

4 EXPERIMENTAL METHODOLOGY

We developed a prototype ontology matching tool in order to assess the quality of the ontology distance function. The next section describes the prototype. Special emphasis is put on the list of parameters to the LRAAM, as these will be investigated in the experimental protocol. The experimental protocol is presented next. As this study is exploratory, the examples used are in their simplest form in order to control all input parameters to better assess the quality of the distance function, given known input ontologies.

4.1 Prototype Ontology Matching Tool

In order to test the LRAAM-based distance function, a tool was developed to receive, transform, and process the input, and determine the similarity score, that is d_{rc}. The tool extends *Neuroph*[4] to realize the LRAAM topology. The tool receives the following input data:

- A process ontology (such as Fig. 1) and two or more service ontologies in order for the tool to construct $r \in R$ and $c \in C$.
- LRAAM-specific input parameters (see also (Ellingsen, 1997)):
  - ϵ: total error to reach before training stops.
  - η: backpropagation learning rate.
  - σ_z: slope of hidden layer activation function $\varphi(\upsilon) = 1/(1 + e^{\sigma_z \upsilon})$, with υ the sum of input to the unit.
  - σ_h: slope of the activation function $\varphi(\upsilon) = 1/(1 + e^{\sigma_h \upsilon})$ of the label part of input/output vector $\mathbf{x}$, being $\mathbf{x}_h$.
  - σ_p: slope of the activation function $\varphi(\upsilon) = 1/(1 + e^{\sigma_p \upsilon})$ of the pointer part of input/output vector $\mathbf{x}$, being $\mathbf{x}_p$.
  - m: freely defined number of hidden layer nodes, with $m < |x|$. Note: It determines the size of the input vector x (see again Fig. 3).

Based on this input data, the tool initializes the following variables, which are described and discussed in the subsequent sections:

[3]The pointer p represents an edge of the graph G.

[4]http://neuroph.sourceforge.net/

- $V_r = \{v_r\}$, the set of vertices of a reference ontology, where v_r is a vertex in the graph representation of reference ontology $r \in R$.
- $V_c = \{v_c\}$, the set of vertices of a compound ontology, where v_c is a vertex of the graph representation of compound ontology $c \in C$.
- $E_{v_r} = \{v'_r\}$, the set of edges in a reference ontology, such that an edge starts from v_r and ends at v'_r.
- $E_{v_c} = \{v'_c\}$, the set of edges in a compound ontology, such that an edge starts from v_c and ends at v'_c.
- $q \leftarrow maxOutDegree(V_r \cup V_c)$, the maximum number of connection slots to include in the input and output vectors of the LRAAM.
- $l_{max} \leftarrow maxLabelLength(V_r \cup V_c)$, the maximum number of $\{-1, 0, 1\}$ values required to encode the labels of the vertices.
- $t_h = q + l_{max}$, the number of $\{-1, 0, 1\}$ values required in the input and output vectors to encode the different vertices.
- $t_p \leftarrow m \cdot q$, the number of real values required in the input and output vectors to encode the connections between vertices (pointers).
- $\mathbf{z}^i_{v_r}$ is the reduced representation of vertex $v_r \in V_r$ at the beginning of a training iteration. It is a vector of m values.
- $\mathbf{z}^o_{v_r}$ is the reduced representation of vertex $v_r \in V_r$ at the end of a training iteration. It is a vector of m values.
- $\mathbf{x}^i_{v_r} = (\mathbf{x}^i_{h,v_r}, \mathbf{x}^i_{p,v_r})$ is the input vector of vertex $v_r \in V_r$ at the beginning of a training iteration, where $\mathbf{x}^i_{h,v_r}$ is a $\{-1, 0, 1\}$ value vector encoding the label and pointer conditions of vertex v_r and $\mathbf{x}^i_{p,v_r}$ is a real vector encoding the pointers outgoing from vertex v_r.
- $\mathbf{x}^o_{v_r} = (\mathbf{x}^o_{h,v_r}, \mathbf{x}^o_{p,v_r})$ is the output vector of vertex $v_r \in V_r$ at the beginning of a training iteration, where $\mathbf{x}^o_{h,v_r}$ is a $\{-1, 0, 1\}$ value vector encoding the label and pointer conditions of vertex v_r and $\mathbf{x}^o_{p,v_r}$ is a real vector encoding the pointers outgoing from vertex v_r.
- $\mathbf{z}^i_{v_c}$ is the reduced representation of vertex $v_c \in V_c$ at the beginning of a training iteration. It is a vector of m values.
- $\mathbf{z}^o_{v_c}$ is the reduced representation of vertex $v_c \in V_c$ at the end of a training iteration. It is a vector of m values.
- $\mathbf{x}^i_{v_c} = (\mathbf{x}^i_{h,v_c}, \mathbf{x}^i_{p,v_c})$ is the input vector of vertex $v_c \in V_c$ at the beginning of a training iteration, where $\mathbf{x}^i_{h,v_c}$ is a $\{-1, 0, 1\}$ value vector encoding the label and pointer conditions of vertex v_c and $\mathbf{x}^i_{p,v_c}$ is a real vector encoding the pointers outgoing from vertex v_c.
- $\mathbf{x}^o_{v_c} = (\mathbf{x}^o_{h,v_c}, \mathbf{x}^o_{p,v_c})$ is the output vector of vertex $v_c \in V_c$ at the beginning of a training iteration, where $\mathbf{x}^o_{h,v_c}$ is a $\{-1, 0, 1\}$ value vector encoding the label and pointer conditions of vertex v_c and $\mathbf{x}^o_{p,v_c}$ is a real vector encoding the pointers outgoing from vertex v_c.

4.2 Experimental Protocol

The experiment is divided into a preliminary and three main phases. In the preliminary phase, a simple process ontology composed of the last two activities from Figure 1 is constructed, namely activities *CreditChecking* and *CreditApproving*. In addition, two simple service ontologies are created. Each service is described by exactly one label. The labels are also $CreditChecking$ and $CreditApproving$. We used Protégé[5] to construct the different ontologies.

The ontologies are loaded into the tool, which then generates new composed ontologies r: $CreditChecking \times CreditApproving$, c_1: $CreditApproving \times CreditChecking$, and c_2: $CreditChecking \times CreditApproving$. As we explore the potential of LRAAM as a classification tool at this time, the experiment is not extended to more than two related activities.

In a further step, the ontologies r and c_i are used to implement the LRAAM to calculate d_{rc_i}, with $i = \{1, 2\}$.

For the analysis, the following variables are defined:

- w: the expected best (winning) service, with $w \in \{1, 2\}$. For the controlled experiment, we expect $w = 2$, as $r \equiv c_2$.
- d_i: distance of service combination c_i, with $i = \{1, 2\}$, and r.
- d_w: distance of expected winning service combination c_w and r
- $\mathfrak{r}_i$: ranking of service combination i out of all combinations.
- $\mathfrak{r}_w$: ranking of expected winning service combination out of all combinations.
- b: 1 if correctly matched, i.e., $w = 2$, 0 otherwise.

In Phase 1, the relationship between the LRAAM input $\epsilon, \eta, \sigma_z, \sigma_h, \sigma_p, m$ and output d_i is analyzed. For each parameter, interval and tick size is fixed to

[5] http://protege.stanford.edu/

explore reasonable, yet limited, parameter combinations within LRAAM's large state space. Tick size is the value of each increment of the parameter value from lowest to largest value in the interval. For example, an interval from 0 to 10, with tick size of 5 would test values in the set $\{0, 5, 10\}$.

All parameter values are thus recombined with each other. To control processing time, one LRAAM instance per combination is executed. A multiple linear regression is conducted to identify the parameters having a significant correlation with d_i. These parameters will be analyzed further in subsequent phases. The other parameters are then fixed heuristically at a value which maximizes b for d_w (i.e., the number of good classifications). At this point, classification performance is not evaluated.

In Phase 2, we further explore the impact of parameters with a significant correlation with d_i. Specifically, the interval is increased and the tick size is decreased. More of LRAAM's state space is thus explored, this time to find optimal parameter values. All resulting parameter values are recombined with each other. Similar to Phase 1, one LRAAM instance per combination is executed. Each variable parameter is heuristically analysed for promising values which maximize b for d_w. Classification performance is still not evaluated.

In Phase 3, 2000 LRAAM runs are executed for the promising values found in Phase 2, which are analysed for classification performance. Specifically, each list is transformed by assigning a ranking $\mathfrak{r}_i$ to d_i. The smallest of two d_i per run is given a ranking $\mathfrak{r}_i = 1$. The other ranking ($\mathfrak{r}_i = 2$) is transformed into a weighted ranking, normalized over the maximum distance spread over all runs, which gives a finer account of the classification performance.

The general success criteria is defined as, $\mu_{\mathfrak{r}_w} < \mu_{\mathfrak{r}_i}, \forall i \neq w$ which in the experiment is $\mu_{\mathfrak{r}_2} < \mu_{\mathfrak{r}_1}$. In other words, the mean ranking of the expected winning service combination c_w must be smaller than the respective means of all other service combinations c_i.

If a list respects the criterium, it is selected and tested for statistical significance. To this end, an independent samples t-test is used, as the sample groups i are independent of each other. Furthermore, as no experiment is conducted on the same subjects before and after some event, a paired t-test is not applicable. If the null-hypothesis ($\alpha = 0.05$) can be rejected, d_w is correctly classified. In that case, the best combination of services (c) to support related business activities (r) could be identified.

5 EXPERIMENTAL RESULTS

In Phase 1, a list of 12,636 d_i is generated, corresponding to 6,318 parameter value combinations. The model summary of the conducted multiple regression is shown in Table 3:

Table 3: Multiple regression model summary.

Predictors	R	R^2	Adj. R^2	Std.Err.
η	0.857	0.735	0.735	0.58527
η & m	0.915	0.838	0.838	0.45722
η & m & σ_h	0.949	0.900	0.900	0.35947
η & m & σ_h & σ_p	0.949	0.900	0.900	0.35935

The predictive capacity of η (learning rate) and m (number of hidden layer nodes) is high as these parameters explain a large part of d's variability ($R^2 = 83.8\%$). Both parameters are promoted to Phase 2. As the other parameters (IO layer binary slope σ_h, IO layer real slope σ_p) do not have a significant impact on d, they are fixed based on suggestions made in (Ellingsen, 1997). We have $\epsilon = 0.15$, $\sigma_h = 6$ (IO layer binary slope), $\sigma_p = 0.5$ (IO layer real slope), and $\sigma_z = 0.5$ (hidden layer slope).

For σ_z, σ_h, and σ_p, the values selected also yield the highest count of correctly matched c in the list (see marked area in Table 4).

Table 4: Parameter combination with highest count of correctly mapped c (framed).

Epsilon	IO layer binary slope	IO layer real slope	Hidden layer slope	Correct matches
0.15	6	0.50	0.50	29
0.20	6	1.50	0.40	26
0.20	7	1.00	0.70	26
0.15	5	1.50	0.70	25
0.25	5	1.00	0.30	25
0.25	7	0.50	0.70	25

In Phase 2, the interval of m and η is increased to cover more of LRAAMs state space. A list of 6,060 d_i is generated, corresponding to 3,030 parameter value combinations. From this list, zones of promising matching performance are identified (Fig. 4 and 5).

The values $\eta = \{0.09; 0.20\}$ and $m = \{22; 42; 105\}$ are fixed for further processing. Consequently, six combination trials of 2,000 runs are executed during Phase 3. The results are shown in Table 5. The classification passed the success criterium, namely, $\mu_{\mathfrak{r}_w} < \mu_{\mathfrak{r}_i}$, in four of the six parameter combinations (in **bold**).

The significance of the results is verified by an independent samples t-test (see Tab. 6). As we can see, we only have a significant classification for the parameter combination (0.20,22), where a very low p-value of 0.002 is obtained. It suggests that the mean

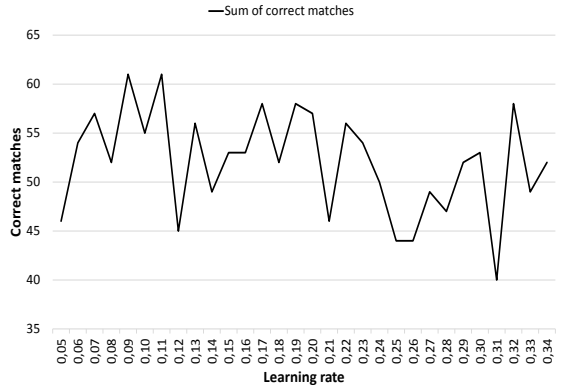

Figure 4: Phase 2: Correct matches for η's tested interval.

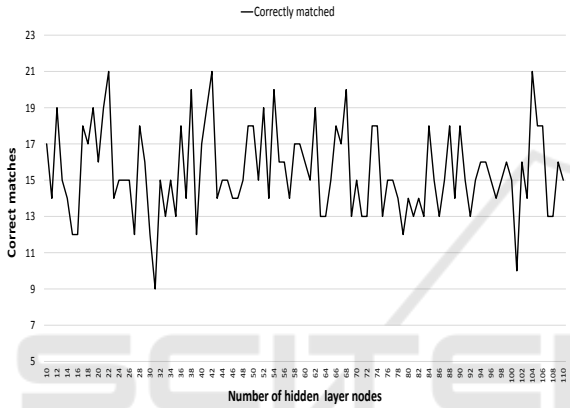

Figure 5: Phase 2: Correct matches for m's tested interval.

ranking of compound ontology $c_2 = 1.89$ is significantly different from the mean ranking of $c_1 = 2.02$. With $\alpha = 0.05$, in this case, the null-hypothesis is thus rejected. In other words, with 2,000 runs, the combination correctly classifies the input. However, SPSS only allows for 2-tailed t-tests, whereas we are interested only in the left tail of the t-distribution. Since the t-test supposes a symmetrical distribution, in this case, the "significant" tail will reflect significance at $p/2 = 0.025$ or below[6]. As $0.002 < 0.025$, the validity of rejection holds.

6 DISCUSSION AND CONCLUSION

In this paper we introduced a service selection and composition approach towards automatic, yet flexible integration of applications wrapped into services. We discussed its necessity for organizations to cope with an increasing number of modularized and decentralized services from which to choose, especially in the context of the proliferation of cloud-based services.

[6]http://www-01.ibm.com/support/docview.wss?uid=swg21476176 (visited May 19, 2015).

Table 5: Classification results for fixed parameter value combinations.

Parameters		**Matches**	
η	m	c_1	c_2 (c_w)
0.09	22	989	**1,011**
0.20	22	949	**1,051**
0.09	42	996	**1,004**
0.20	42	1,005	995
0.09	105	974	**1,026**
0.20	105	1,042	958

Table 6: t-test for significance of $\mu_{\mathfrak{r}_w} < \mu_{\mathfrak{r}_i}$.

Parameters		**Levene's test**		**t-test for equality of μ**		
η	m	F	Sig.	t	df	Sig.(2-tailed)
0.09	22	0.792	0.374	1.122	3,998	0.262
0.20	22	2.232	0.135	3.025	3,998	0.002^*
0.09	42	0.368	0.544	0.587	3,998	0.557
0.09	105	0.000	0.991	1.148	3,998	0.251

* H_0 rejected.

As other authors (e.g., (Born et al., 2007; Fensel et al., 2011)), we believe that ontologies lend themselves naturally as building blocks. The idea is that business experts may use ontologies to describe corporate business activities, on the one hand, and service providers use ontologies to describe their service offerings, on the other hand, so machines can compare them automatically. It is a reasonable conjecture that an ontology describing a credit checking activity uses "similar" entities and relations as the needed credit checking service. In our example, both possibly contain the entities *Person* and *Customary credit register*, including a relation *hasEntry* between them. In turn, an ontology describing a weather forecast service most probably does not contain an entity *Customary credit register*.

With the implemented LRAAM oracle, we conducted experiments to verify similarity of services. The results obtained are limited, as only one pair of parameters in the space investigated yielded significant results. Consequently, no general conclusions may be drawn from these experiments. This however, does not imply that LRAAM should not be considered altogether to perform service selection. Indeed, we feel that the implemented tool needs further development and pruning to increase its performance. The following points caught our attention:

- As part of the sigmoid activation function, a higher impact of σ on d was anticipated. This could however not be shown and warrants further investigation.
- The significant parameter combination $(0.2; 22)$

leads to a ratio $m/n = 22/151 = 0.15$. Based on (Ellingsen, 1997), a good classification performance was expected at a higher ratio of ≈ 0.25.

- Ontologies c and r contain the same labels by default, namely those representing the sequence ontology, which we use to calculate the Euclidean distance. In the experiment, only two labels are added to it to construct c, respectively r. These labels may not have sufficient impact to significantly alter the LRAAM's rather volatile steady state. In other words, distinctiveness, which is sought, gets lost (see next point).
- We do not think that the current LRAAM implementation exhibits a sufficiently trustworthy steady state. Instead of initializing the input vector x with values from the interval $[-1, 1]$, similar to (Al-Said and Abdallah, 2009), it may be beneficial to use a smaller interval, e.g., $[-0.5; 0.5]$, to decrease potential variability, such as described in (Sperduti, 1993). Nevertheless, in light of the similarity already given between the two particular input strings, the results seem promising.
- Narrowing down the LRAAM's state space is time-consuming. However, further intervals or parameters need to be explored, also combined with the above mentioned adaptations. One further parameter may be the number of consecutive times the LRAAM's establishes a steady state (expressed in $\mathbf{Z}$) based on the error tolerance ϵ, before it is chosen to calculate d. It would increase confidence in the validity of the steady state.

The above list mainly points towards improvements of the LRAAM's existing architecture, i.e., tuning the activation function, varying the ratio of input and hidden layer, etc. However, a further route to enhancing classification performance may reside in changing the LRAAM's architecture altogether. Specifically, the idea would be to couple the LRAAM with the *Deep Learning* (DL) approach. The latter represents an ANN with up to 20 hidden layers (instead of only one). In total, such an ANN may contain tens or hundreds of thousands of units. Following (LeCun et al., 2015), a DL system can implement extremely intricate functions of its inputs that are simultaneously sensitive to smallest variations. It exploits the property that many input signals are compositional hierarchies, in which higher-level features are obtained by composing lower-level ones. The authors continue to state that similar hierarchies exist in speech and text from sounds to phones, phonemes, syllables, words and sentences. Clearly, this may prove beneficial to our approach. Finally, for DL, poor local minima are rarely a problem. Notwithstanding the initial conditions, i.e., the initialized random values, the system nearly always reaches high quality solutions.

In Figure 6, a possible realization is sketched. Instead of initializing the label part of LRAAM's input vector with binary information, one could use the Deep Learning approach to preprocess those concept labels.

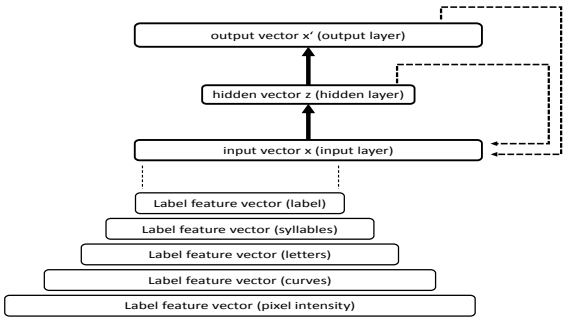

Figure 6: Deep Learning (DL) and LRAAM. Concept labels are preprocessed by DL before participating in the input vector x.

Summarizing, in this paper, we explored ontology matching as a means towards dynamic service integration. Thereby, we limited our focus on ontological descriptions about what a service actually does. We followed the premise that the richer a service is described, the better it can be evaluated and selected, provided a holistic (micro-semantic) matching method.

REFERENCES

Al-Said, G. and Abdallah, M. (2009). An arabic text-to-speech system based on artificial neural networks. *Journal of Computer Science*, 5(3):207–213.

Antoniou, G. and van Harmelen, F. (2008). *A Semantic Web Primer*. The MIT Press, Cambridge Massachusetts, 2 edition.

Antoniou, G. and van Harmelen, F. (2009). Ontology web language: Owl. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 91–110. Springer.

Atencia, M., Euzenat, J., Pirrò, G., and Rousset, M.-C. (2011). Alignment-based trust for resource finding in semantic p2p networks. In *The Semantic Web–ISWC 2011*, pages 51–66. Springer.

Blank, D., Meeden, L. A., and Marshall, J. B. (1992). Exploring the symbolic/subsymbolic continuum: A case study of RAAM. In *The Symbolic and Connectionist Paradigms: Closing the Gap*, pages 113–148. Erlbaum.

Born, M., Drumm, C., Markovic, I., and Weber, I. (2007). SUPER - raising business process management back to the business level. *ERCIM News*, 2007(70).

Bughin, J. and Chui, M. (2010). The rise of the networked enterprise: Web 2.0 finds its payday. *McKinsey quarterly*, 4:3–8.

Chan, S. W. K. (2003). Dynamic context generation for natural language understanding: A multifaceted knowledge approach. *IEEE Transactions on systems, man, and Cybernetics - Part A: Systems and Humans*, 33(1):23–41.

Cisco (2014). Cisco global cloud index: Forecast and methodology: 2013 - 2018. White paper, Cisco Systems Inc.

de Gerlachey, M., Sperdutiz, A., and Staritaz, A. (1994). Using labeling raam to encode medical conceptual graphs. *NNESMED'94 proceedings*.

Diallo, G. (2014). An effective method of large scale ontology matching. *Journal of Biomedical Semantics*, 5:44.

Eder, J. and Wiggisser, K. (2007). Detecting changes in ontologies via DAG comparison. In *Lecture Notes in Computer Science 4495*, pages 21–35.

Ellingsen, B. K. (1997). Distributed representations of object-oriented specifications for analogical mapping. Technical report, Citeseer.

Erl, T. (2004). *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice Hall PTR.

Erl, T., Chelliah, P., Gee, C., Kress, J., Maier, B., Normann, H., Shuster, L., Trops, B., Utschig, C., Wik, P., and Winterberg, T. (2014). *Next Generation SOA: A Concise Introduction to Service Technology & Service-Orientation*. The Prentice Hall Service Technology Series from Thomas Erl. Pearson Education.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer.

Euzenat, J. and Shvaiko, P. (2013). *Ontology Matching*. Springer, 2 edition.

Fensel, D. and Bussler, C. (2002). The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137.

Fensel, D., Facca, F. M., Simperl, E., and Toma, I. (2011). *Semantic web services*. Springer Science & Business Media.

Frank, R. and Cartwright, E. (2013). *Microeconomics and Behaviour*. McGraw Hill.

Hinton, G., McClelland, J., and Rumelhart, D. (1986). *Distributed representations*, volume 1, pages 77–109. MIT Press.

Hoang, H. H., Jung, J. J., and Tran, C. P. (2014). Ontology-based approaches for cross-enterprise collaboration: A literature review on semantic business process management. *Enterprise Information Systems*, 8(6):648–664.

Hoang, H. H. and Le, M. T. (2009). Bizkb: A conceptual framework for dynamic cross-enterprise collaboration. In Nguyen, N. T., Kowalczyk, R., and Chen, S.-M., editors, *ICCCI*, volume 5796 of *Lecture Notes in Computer Science*, pages 401–412. Springer.

Hussain, M. A. and Mastan, M. (2014). A study on semantic web services and its significant trends. *IJCER*, 3(5):234–237.

Izza, S. (2009). Integration of industrial information systems: from syntactic to semantic integration approaches. *Enterprise Information Systems*, 3(1):1–57.

Kale, V. (2014). *Guide to Cloud Computing for Business and Technology Managers: From Distributed Computing to Cloudware Applications*. Taylor & Francis.

Kotis, K., Vouros, G., and Stergiou, K. (2006). Towards automatic merging of domain ontologies: The hcone-merge approach. *Journal of Web Semantics (JWS)*, 4:60–79.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lehaney, B., Lovett, P., and Shah, M. (2011). *Business information systems and technology: a primer*. Routledge.

Li, W., Raskin, R., and Goodchild, M. F. (2012). Semantic similarity measurement based on knowledge mining: An artificial neural net approach. *International Journal of Geographical Information Science*, 26(8):1415–1435.

Ludolph, H., Kropf, P., and Babin, G. (2011). Softw*I*re integration - an onto-neural perspective. In Babin, G., Stanoevska-Slabeva, K., and Kropf, P., editors, *E-Technologies: Transformation in a Connected World - 5th International Conference (MCETECH 2011). Les Diablerets, Switzerland, January 23-26, 2011, Revised Selected Papers*, number 78 in Lecture Notes in Business Information Processing, pages 116–130. Springer.

Otero-Cerdeira, L., Rodríguez-Martínez, F. J., and Gómez-Rodríguez, A. (2015). Ontology matching: A literature review. *Expert Systems with Applications*, 42(2):949–971.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Rahm, E. (2011). Towards large-scale schema and ontology matching. In *Schema matching and mapping*, pages 3–27. Springer.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4).

Shvaiko, P. and Euzenat, J. (2013). Ontology matching: state of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):158–176.

Sperduti, A. (1993). On some stability properties of the lraam model. Technical report, International Computer Science Institute.

Zdravković, M., Trajanović, M., and Panetto, H. (2014). Enabling interoperability as a property of ubiquitous systems: towards the theory of interoperability-of-everything. In *4th International Conference on Information Society and Technology, ICIST 2014*, volume 1, pages 240–247, Kopaonik, Serbia.