A Suggestion-Based RDF Instance Matching System

Mehmet Aydar and Serkan Ayvaz

***Abstract*—This paper presents a semi-automatic recommendation-based instance matching system using RDF graph data. Based on a graph node similarity algorithm, our instance matching system detects instance nodes with similarities higher than an input threshold value and returns to the user the subject node pairs. The system merges a matched node pair when the user confirms the matched nodes in the results. After a merge, the merged node is also considered as an entity for the following candidate pair generation cycle. The procedure continues until no new matching candidate pairs are recommended by the algorithm and there is no more feedback provided by the user.**

***Index Terms*—Instance Matching, RDF, Semantic Web, Similarity Metrics.**

I. Introduction

In this study, we utilize an RDF entity similarity algorithm for matching instances that may be merged if confirmed by the user. Our assumption is that two graph entities are similar if their neighbor entities are also similar. Our semi-automatic instance algorithm runs in iterations with the input from the user. At each consequent iteration, the algorithm generates more precise results based on the input from the user. In addition, our technique reduces the size of the RDF graph since we merge the same or very similar RDF nodes. As a result, the process reduces the complexity of the similarity algorithm at each iteration.

A. Semantic Web and RDF

Resource Description Framework (RDF) [1] is a general purpose language, for representing information in Semantic Web [2] in a way that the meaning (or semantics) is unambiguous to a machine or software process. RDF describes resources through statements in the form of (subject, predicate, object) expressions which are known as triples.

The development of Semantic Web technologies has led to significant progress including explicit semantics with data in the Web in recent years. As increasing number of organizations adopt Semantic Web technologies, publishing data in a standard model and interlinking the data available on the Web using Semantic Web technologies provide a Web of data that is machine accessible and can be utilized by applications through semantic queries.

The collection of inter-linked datasets on the Web is also referred to as Linked Data [3]. With the contribution of Linked Open Data along with several other Semantic Web projects, the structured data available in the Semantic Web have been increasing exponentially. Many datasets in various domains such as publications, life sciences, media, social web, geography have been incorporated into the Linked Open Data.

Manuscript received February 5, 2017; revised April 15, 2017.

M. Aydar is with the Department of Computer Science, Kent State University, Kent, OH 44240 USA (e-mail: maydar@ kent.edu).

S. Ayvaz is with Department of Software Engineering, Bahcesehir University, Besiktas 34353, Istanbul, Turkey (Corresponding author; e-mail: serkan.ayvaz@eng.bau.edu.tr).

B. Data Mapping and Linking

Data mapping [4] is the process of creating the linkages and relations between data elements of distinct data models. Data mapping creates connections between different data elements. The connectivity of the data elements increases data interoperability and data reusability while reducing the redundancy. Moreover, it is a key task for data integration processes including data transformation, data lineage analysis, discovery of new data details within connected data sources, consolidation of multiple data sources into a single data source, etc. Furthermore, data mapping is needed for standardization of the data. For instance, healthcare institutions often need to map their local data to an accepted medical standard such as ICD-9 [5] or SNOMED CT [6] to be able to share their data with other medical facilities.

In essence, the task of data linking is connecting semantically related instances from multiple data sources. For our purposes, we use the term instance matching as finding the semantically matching instances between multiple RDF graphs. The matched instances do not necessarily have to be identical or equivalent. They might also be hierarchically related with subset or superset relations.

C. Instance Matching for Semantic Interoperability

Instance matching is an essential task in achieving semantic interoperability on the Semantic Web. As the amount of publicly available heterogenic data on the Semantic Web grows continually, the applications require creating more and more linkages between the data sources.

For instance, in [7], the authors present a method for the translation of data models from one format to another. The Fig. 1 below shows how their system works.

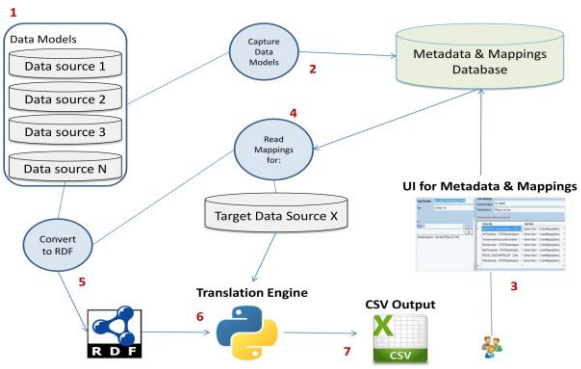

Fig. 1. translation of instance data (taken from [7]).

DOI: 10.7763/IJCTE.2017.V9.1170

As shown in the figure, the approach defined a number of steps required to perform the translation to support heterogeneous data interoperability. In step 3 of the approach, the mapping and the translation rules between various schemata are manually defined a priori from a domain expert.

The manually defining the mappings may result to time-consuming and costly work in the process due to the size the data. The RDF instance matching system introduced in this study contains a semi-automatic instance matching framework to help experts find linkages between different data elements.

D. Semi-automatic Instance Matching Technique

The computation of entity similarity is essential for the instance matching task as our instance matching system utilizes entity similarities to link the same or similar real-world objects. Our system uses a pairwise entity similarity algorithm for RDF graph data as explained in section IV. As the similarity algorithm generates the entity similarity results, our system processes the results and it returns the subject node pairs with a similarity score higher than a threshold. If the user confirms a matched node pair in the results, the system merges the nodes. After a merge, the merged node is also considered as an entity for the following candidate pair generation cycle.

The system then reruns the similarity algorithm with the merged RDF node pairs. Based on the common predicates and neighbor similarity, a merged node can be matched with another instance and presented to the user as a candidate pair. This process continues until there is no more feedback from the user. Each time the similarity algorithm produces more accurate results with the input from the user. The size of the input RDF graph data is reduced by merging process, yielding less complexity each time.

II. Contribution and Outline

This study investigates the problem of discovering the linkages between semantically related entities that could be classified as the same entity within and among different data sources. Thus, the same or very similar entities can be represented by a single entity. In the literature, this problem has been studied by the research community as the task of instance matching or concept matching. In this context, we consider this problem an instance matching of RDF entities, as we represent the data instances and data model details in an RDF model. Our approach is semi-automatic: it utilizes a pairwise graph node similarity computation algorithm for finding similar entities for presenting the similar entities to the domain experts.

Our main contributions:

- We present a suggestion-based semi-automatic instance matching system utilized for the RDF representation of data that contributes in the information translation process.
- We use neighborhood similarity idea to infer possible connections between the entities, such that if two entities are matched, then other nodes with similar predicates and in similar neighborhoods are considered.
- We merge the matched entities and run the process in iterations, and producing more accurate results and yielding less complexity after each iteration.

The paper is organized as follows. We discuss the application of instance matching in an RDF representation of data elements. Subsequently, we review the computation of entity similarity that is used for the matching. Then, we describe the user interaction for semi-automatic instance matching process. The subsequent section presents the results of the evaluations. In the following section, the related work is reviewed and followed by our conclusion.

III. Instance Matching in RDF Graph

The source data for an RDF graph may exist in heterogeneous data sources with different formats and data models. In this work, we assume that the data elements are represented in RDF: i.e., the instance data, the data dictionary elements that belong to the instance data and the mapping between them are represented in RDF. For the data sources that do not have a pre-defined data dictionary in place, a summary graph generation algorithm as in [8] can be exploited to extract the data dictionary elements. The main data dictionary classes can be linked to the instance data elements with the rdf:type [9] predicate. The RDF relations allow linking the data elements from different systems, constituting an extensive and connected RDF graph.

In such a data ecosystem, it is common to have redundant instances and concepts between different data sources. Thus, the instance matching technique explained in this study covers both de-duplication and data concepts matching. Merging the redundant nodes helps to reduce the size of the dataset. Also, linking the concepts between diverse data models assists in the information translation process and semantic data interoperability of different systems.

IV. RDF Entity Similarity for Instance Matching

In the studies [8], [10], we introduced an effective algorithm for the computation of pairwise graph node similarity using graph locality, neighborhood similarity, and the Jaccard measure. In the similarity algorithm, the computation of entity similarity is studied as a pairwise RDF graph node similarity problem. The algorithm is based on an efficient graph node similarity metric. Our instance matching system uses this RDF entity similarity algorithm for pairing entities.

An RDF entity is described through a set of predicates, the collection of literal neighboring nodes that it references and the neighbor nodes with which it interacts. The predicates of the subject nodes are treated as the dimensions of the entities. We utilize the common descriptors within the Jaccard measure context when calculating the similarity of an RDF node pair, along with the similarities of their neighbors. Thus, the direct similarities of the entities are taken into account along with the similarities of the neighbors with which they interact.

Each descriptor of an RDF node may have a different impact in the similarity calculation, in other words each

descriptor has a different importance weight. Therefore, we presented an importance weighting metric for the descriptors of the RDF nodes and enhanced the similarity metric by incorporating the auto-generated importance weights of the descriptors.

V. User Interaction

The size of data in semantic interoperability tasks often requires an automated instance matching method. Nonetheless, fully automated techniques can be error prone. Therefore, a semi-automated instance matching technique with user interactions yields more accurate results. Since our goals include matching the instances belonging to the heterogeneous data sources, our semi-automatic instance matching system, allows the user to provide initial matches between the source and target graph elements. The system then runs the entity similarity algorithm introduced in section IV. After the entity similarities converge, it follows the steps below:

- The system extracts the subject IRI node pairs which have similarities higher than a user defined and configurable parameter (threshold), and then presents them to the user.
- A subject node pair that is presented to the user is denoted by ($s1$,$s2$) where *s1* and $s2$ are two subject IRI nodes having similarity greater than the defined threshold. Our system merges these two nodes if their match is approved by the user. The merged node is considered as a single subject node which is denoted by [$s1$,$s2$], that retains all the predicates all from both $s1$ and $s2$.
- Our system then checks the common object nodes of $s1$ and $s2$, and generates ($p1$,$p2$) as a candidate instance pair if both *s1* and *s2* are connected to a common object node by $p1$ and $p2$ correspondingly.
- In addition, the system checks the common predicates of $s1$ and $s2$, and generates ($o1$, $o2$) as a candidate instance pair if both $s1$ and $s2$ correspondingly connects to the object nodes $o1$ and $o2$ by a common predicate.
- The instance matching candidates ($p1$,$p2$) and ($o1$, *o2*) are then presented to the user and merged if their match is approved by the user. The merged entities are denoted by [$p1$,$p2$] and [$o1$,$o2$].
- The system then reruns the RDF entities similarity algorithm based on the new RDF graph generated by the merged graph entity pairs.
- Our system repeats the steps explained above until the there is no new matching pairs generated by our algorithm and no more feedback from the user.

An example of how our system handles the instance matching and merging process is shown in Fig. 2. In the figure, the nodes $v1$, $v2$, $v3$, $v4$ and the predicates $p1$, $p2$, $p3$ belong to the source graph while the nodes $v2$, $v5$, $v6$ and the predicates $p3$, $p4$ belong to the target graph. Our goal is to find the matching instances between the source and the target graph.

At first, the RDF entities similarity algorithm runs and our instance matching system generates the first instance matching candidates based on the results of the similarity algorithm. As shown in Fig. 2, our system pairs the subject nodes ($v1$, $v5$) as a candidate instance matching pair and presents them to the user at phase 1. The user approves that the candidates match and the subject nodes $v1$, $v5$ are merged to get [$v1$,$v5$].

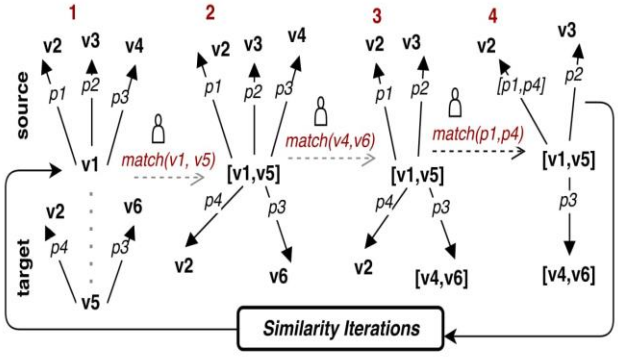

Fig. 2. Instance matching process.

Then, on phase 2, the common predicates of the merged node [$v1$,$v5$] are checked by the algorithm. As the new node [$v1$,$v5$] connects to the subject nodes $v4$ and $v6$ by the common predicate *p3*, our system presents ($v4$,$v6$) to the user, and it gets [$v4$,$v6$] once the user approves that they match.

On phase 3, our system checks the common object nodes of the new node [$v1$,$v5$]. As it is clear that [$v1$,$v5$] connects to a common object node *v2* with the predicates *p1* and *p4*, the algorithm presents the predicates pair ($p1$,$p4$) to the user, and it merges them to get [$p1$,$p4$] upon approval by the user.

At phase 4, comparing to phase 1, the source and target graphs are merged, and the graph size in total is reduced by *40%* (from five triples to three triples). As the instance matching system runs in iterations, in the next iteration, the output graph of phase 3 becomes input in phase 1. The iterations continue until the optimum instance matching pairs are obtained.

VI. Evaluation

In empirical evaluations, the following datasets were used: a subset of DBpedia [11] and a subset of SemanticDB [12]. SemanticDB is a Semantic Web data repository developed by the Cleveland Clinic for Clinical Research and Quality Reporting. To evaluate the effectiveness of the instance matching algorithm, we generated a validation dataset by replicating the original dataset and syntactically changing the names of the instances. We transformed instance names in the validation dataset using a specific naming pattern. The original dataset was considered as the source, and the validation dataset was as the target in the instance matching step. The instance node naming pattern was used for validation. In summary, the instance matching evaluation with DBpedia as the source dataset included 90 triples with *60* distinct subject and predicate nodes. The algorithm semi-automatically matched 100% of the nodes to a target graph node. The algorithm achieved 85% accuracy and generated *20* instance matching candidates. The evaluations

with SemanticDB as the source dataset contained 2500 triples with 520 distinct subject and predicate nodes. 86% of the nodes were matched to a target graph node semi-automatically. The accuracy of the algorithm was 95% and 310 instance matching candidates were generated the algorithm.

VII. Related Work

In the literature, there has been much research in the subject of ontology mapping. The terminology used for defining the problem has varied such as matching, alignment, merging, articulation, fusion, integration, morphism, etc. Inherently, many tools and methods have emerged in the field.

Many of these approaches are solely based on the use of string similarity mechanisms for finding matching entities between two ontologies [13], [14]. Approaches that have studied the ontology mapping problem from the schema matching perspective in the context of data integration have also been explored since schemata can be considered as ontologies with restricted relation types. These include [15], [16].

Some others have suggested asking the user for feedback, as in our work, to perform the mapping generation interactively by providing proper visualization to support the decision. We think this is a useful feature for generating high-quality links. However, in this work we minimize the need for user feedback to reduce the load on the users.

Doan et al. [17] provided a system, GLUE, which employs learning techniques to semi-automatically find mappings between two given ontologies. For each concept in one ontology, their framework uses a multi-learning strategy to predict a similar concept in the other using probabilistic definitions of several similarity measures. GLUE calculates a joint probability distribution to measure the overlap between two input sets. The authors offer two learners: a content learner and a name learner for learning information such as the word frequencies, instance names and value formats. The content learner uses Naive Bayesian learning, a text classification method, for the instance content whereas the name learner uses the full name instead of its content. Then they combine the predictions of the two learners and assign weights to them using a meta-learner. Additionally, they use a technique, relaxation labelling, which gives labels to nodes of a graph, based on a set of constraints. Similar to their system, we also propose a machine learning framework. In contrast to them, we don't employ an active learning technique, which relies on the quality of the training dataset.

Jain *et al.* offered a framework called BLOOMS [18] and later an improved version under BLOOMS+ [19]. They propose a metric to determine which classes to align between two ontologies and a technique for using contextual information to support the alignment process. However, they rely on existence of a human-generated upper ontology and the concept categorization in the form of a tree structure. Our approach does not rely on an upper ontology, as we think these assumptions are problematic since the quality of the mapping strictly depends on the categorization of the concept by humans, and any potential categorization issue in the upper ontology will have an impact on the whole context.

The notion of instance matching has also been studied by the Semantic Web community. With an ontology instance matching perspective, some research studies have investigated comparing instances based on the properties and roles. However, they primarily focus on the ontology mapping [20], [21] or ontology population [22] tasks.

On the other hand, [23] studies an automatic instance matching problem in RDF graphs with the focus on property weights, where property weights yield precedence to properties that make the instances more unique. The similarity metric utilized in this work also employs auto-generated property weights, which is a similar notion to the term frequency-inverse document frequency (tf-idf) [24], [25]. Our work is also different in the sense that it is semi-automatic, allowing for user feedback.

A matching algorithm called similarity flooding (SF) is proposed in [8]. SF matches two directed and labeled graphs to produce a multi-mapping of analogous nodes.

For the similarity computation, SF relies on the intuition that elements of two graphs are similar when their adjacent elements are similar, exploiting the neighboring structure of a concept map, the semantic meaning of the content of the graph node and the labels of the relations between the nodes.

In SF, the similarity of the node pairs starts either with a string similarity between the content of the nodes or with a similarity of 1. It then propagates the initial similarity of any two nodes through the graphs. The algorithm runs in multiple iterations until the similarity values are converged, or until a pre-defined maximum number of iterations. SF also does not distinguish between a schema node and the instance data node.

In SF, filters are utilized to select the best mappings. The mappings are then manually reviewed. The accuracy of the algorithm is measured by the estimated human labor savings obtained utilizing the algorithm for the matching tasks. In our work, we make an assumption for similarity computation like that of SF, that the nodes connected to similar neighbors with similar predicates are similar. We also run our similarity algorithm in iterations, and we do not distinguish between the data model elements and the instance data elements like in SF. However by the end of the iterations, we suggest the similar node pairs to the user for matching, and we merge the triples of the approved matched nodes. Consequently, we suggest matching of the nodes and predicates based on the common predicates and neighbors of the already matched nodes, and we rerun the similarity iterations. In this sense, our technique requires more user interactions but the consequent similarity iterations produces more accurate results assuming the user provides accurate feedback.

VIII. Conclusion

In this study, we provided an instance matching system using RDF graphs. Our instance matching technique is semi-automatic and does not distinguish between the instance data elements and the data model elements. The system makes use of an efficient similarity algorithm, and it makes smart suggestions to the user by utilizing the neighborhood concept to infer possible connections between the graph

entities for finding the linkages between different data elements. The linkages found can further be utilized in a data translation framework to support data interoperability. Additionally, we performed exploratory evaluations that demonstrated significant results in matching similar graph elements.

ACKNOWLEDGMENT

A special note of thanks to Prof. Austin Melton for his invaluable help and guidance during the study. Also, the authors would like to thank the Cleveland Clinic Cardiology Application Group members for their valuable feedback.

REFERENCES

[1] G. Klyne and J. J. Carroll, "Resource description framework (RDF): Concepts and abstract syntax," 2006.
[2] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, "The semantic web," *Sci. Am.*, vol. 284, no. 5, pp. 28–37, 2001.
[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.
[4] Wikipedia. (2017). *Data Mapping*. [Online]. Available: https://en.wikipedia.org/wiki/Data_mapping
[5] C. for D. Control and Prevention. *ICD - ICD-9 - International Classification of Diseases, Ninth Revision*. [Online]. Available: http//www.cdc.gov/nchs/icd/icd9cm.htm
[6] U. S. N. L. of Medicine. *SNOMED Clinical Terms® (SNOMED CT®)*. [Online]. Available: http//www.snomed.org/snomed-ct
[7] M. Aydar and A. C. Melton, "Translation of instance data using RDF and structured mapping definitions," in *Proc. 14th International Semantic Web Conference ISWC*, 2015.
[8] S. Ayvaz, M. Aydar, and A. C. Melton, "Building summary graphs of RDF data in semantic web," in *Proc. 2015 IEEE 39th International Computer Software and Applications Conference (COMPSAC)*, 2015.
[9] D. Brickley and R. V. Guha, "RDF vocabulary description language 1.0: RDF schema," 2004.
[10] M. Aydar, S. Ayvaz, and A. C. Melton, "Automatic weight generation and class predicate stability in RDF summary graphs," 2015.
[11] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," Springer, 2007.
[12] C. D Pierce, D. Booth, C. Ogbuji, C. Deaton, E. Blackstone, and D. Lenat, "SemanticDB: A semantic Web infrastructure for clinical research and quality reporting," *Curr. Bioinforma.*, vol. 7, no. 3, pp. 267–277, 2012.
[13] D. Spohr, L. Hollink, and P. Cimiano, "A machine learning approach to multilingual and cross-lingual ontology matching," *The Semantic Web–ISWC 2011*, pp. 665–680, 2011.
[14] G. Stoilos, G. Stamou, and S. Kollias, "A String Metric for Ontology Alignment," *The Semantic Web–ISWC 2005*, Springer, pp. 624–637, 2005.
[15] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic schema matching with cupid," *VLDB*, vol. 1, pp. 49–58, 2001.
[16] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm and its application to schema matching," in *Proc. 2002 18th International Conference on Data Engineering*, 2002, pp. 117–128.
[17] H. Bohring and S. Auer, A.-H. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Ontology matching: a machine learning approach," *Handb. Ontol. STAAB STUDER REds Int. Handb. Inf. Syst. Springer Verl. Berl.*, pp. 385–404, 2004.
[18] P. Jain, P. Hitzler, A. P. Sheth, K. Verma, and P. Z. Yeh, "Ontology Alignment For Linked Open Data," *The Semantic Web–ISWC 2010*, Springer, pp. 402–417, 2010.
[19] P. Jain *et al.*, "Contextual ontology alignment of lod with an upper ontology: A case study with proton," *The Semantic Web: Research and Applications*, Springer, pp. 80–92, 2011.
[20] A. Isaac, L. Van Der Meij, S. Schlobach, and S. Wang, "An empirical study of instance-based ontology matching," Springer, 2007.
[21] C. Wang, J. Lu, and G. Zhang, "Integration of Ontology Data through Learning Instance Matching," in *Proc. International Conference on Web Intelligence*, 2006, pp. 536–539.
[22] S. Castano, A. Ferrara, S. Montanelli, and D. Lorusso, "Instance Matching for Ontology Population," *SEBD*, pp. 121–132, 2008.
[23] M. H. Seddiqui, R. P. D. Nath, and M. Aono, "An efficient metric of automatic weight generation for properties in instance matching technique," *Int. J. Web Semantic Technol.*, vol. 6, no. 1, p. 1, 2015.
[24] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM J. Res. Dev.*, vol. 1, no. 4, pp. 309–317, 1957.
[25] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *J. Doc.*, vol. 28, no. 1, pp. 11–21, 1972.

Mehmet Aydar was born in Iskenderun, Turkey in 1986. He took his bachelor degree in computer engineering in 2005 from Bahcesehir University, Istanbul, Turkey. He then took his master's degree in computer technology in 2008 from Kent State University in OH, USA. He also received his Ph.D degree in computer science from Kent State University in 2015, OH, USA.

He worked as a PLC programmer between 2005-2006 in Hipertech LTD, Istanbul, Turkey. He was a graduate assistant between 2007-2008 in Kent State University, OH, USA. Between 2008 and 2011 he worked as a programmer/analyst in visual evidence LLC company, Cleveland, OH. He then worked in Cleveland Clinic Heart and Vascular Institute as a senior system analyst between 2011 and 2016. He is currently the owner of multiple e-commerce web. His research interests and previous publications are in the field of semantic web and its applications in healthcare and life sciences and data mining.

Serkan Ayvaz received his bachelor's degree in mathematics and computer science in 2006 from Bahçeşehir University Istanbul, Turkey. Later, he received his master's degree in technology with specialization in computer technology from Kent State University in 2008. He completed his Ph.D. in computer science at Kent State University in 2015. He has over 8 years of industry work experience in the USA, most recently as lead systems analyst at eResearch Department at the Cleveland Clinic Foundation between 2011 and 2016. In his role, he served on multidisciplinary research teams focusing on medical research projects. Prior to joining the Cleveland Clinic, he had worked as a software engineer at Hartville Group for three years.

He is currently a faculty member at the Department of Software Engineering and serves as the coordinator of the big data analytics and management graduate program at Bahcesehir University.

His research interests include semantic searches, machine learning and scalable knowledge discovery in Big Data Semantic Web and its applications, particularly in healthcare and the life sciences.