

HInT: Hybrid and Incremental Type Discovery for Semantic Graphs

Alexander Miraka¹, Nikos Kardoulakis², Kenza Kellou-Menouer³, Georgia Troullinou², Zoubida Kedad⁴, Dimitris Plexousakis² and Haridimos kondylakis^{2,*}

¹Computer Science Department, University of Crete, Heraklion, Greece

²FORTH-ICS, Heraklion, Greece

³ETIS Lab, ENSEA, Cergy, France

⁴Univ. of Versailles - Paris Saclay, France

Abstract

The rapid proliferation of the semantic web has led to the emergence of many weakly structured and incomplete data sources. On these sources typing information might be partially or completely missing. On the other hand, type information is essential for a number of tasks such as query answering, integration, summarization, and partitioning. Existing approaches for type discovery, either completely ignore type declarations available in the dataset (implicit type discovery approaches), or rely only on existing types, in order to complement them (explicit type enrichment approaches). In this demonstration, we present *HInT*, the first *incremental* and *hybrid* type discovery system for RDF datasets. To achieve this goal *HInT* identifies the patterns of the various instances, and then indexes and groups them to identify the types. Besides discovering new types, *HInT* exploits type information if available, to improve the quality of the discovered types by guiding the classification of the new instance in the correct group and by refining the groups already built.

Keywords

Semantic Graphs, Schema Discovery, RDF

1. Introduction

The proliferation of the semantic web and the explosion of the information now available as linked data have led to many weakly structured, irregular, and incomplete data sources, where declarations on the schema as primitives might be incomplete or missing. On the other hand, type information is crucial for a number of tasks such as federated query answering, data integration, summarization, and data partitioning.

As such, several works have focused in the past on type discovery. Based on our recent survey on semantic schema discovery [1], existing approaches proceed in two completely different ways. The *implicit type discovery* approaches [2, 3, 4] rely on the analysis of the structure of the

ISWC-Posters-Demos-Industry 2022 (International Semantic Web Conference (ISWC) 2022: Posters, Demos, and Industry Tracks)

*Corresponding author.

✉ csd3804@csd.uoc.gr (A. Miraka); nickkard@ics.forth.gr (N. Kardoulakis); kenza.kellou.menouer@gmail.com (K. Kellou-Menouer); troulin@ics.forth.gr (G. Troullinou); Zoubida.Kedad@uvsq.fr (Z. Kedad); dp@ics.forth.gr (D. Plexousakis); kondylak@ics.forth.gr (H. kondylakis)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

instances and completely ignore schema declarations even when they are partially provided in the data source. Existing algorithms for implicit type discovery employ clustering algorithms such as k-means variations, hierarchical clustering or DBScan. However, the number of clusters is not usually known, whereas most of the algorithms require exhaustive comparisons between the instances. On the other hand the *explicit type enrichment* approaches [5, 6, 7, 8] rely on the schema declarations to complement or enrich them. However, these approaches cannot process sources that have little or no schema information. Finally, existing approaches do not natively support incrementality. Each time new information is added, the types are recomputed from scratch, and previously computed information is completely ignored. However, RDF KBs are constantly updated and evolved [9, 10] .

In this paper, we move beyond the state of the art by demonstrating *HInT*, the first incremental and hybrid type discovery system for RDF datasets. *HInT* exploits the structure of the instances in order to discover typing information (as implicit type discovery approaches do), but also when typing information is available for some instances, it also takes it into consideration in order to guide and improve the type discovery process (as explicit schema discovery approaches do). Our approach first identifies the pattern of a given instance, then assigns the instance to the groups with similar patterns, using Locality Sensitive Hashing, and finally identifies the ones sharing the same type. We reduce instance processing to pattern processing, where each instance is treated independently, whereas our approach is incremental and suitable for large data sets. In addition, our approach enables type discovery in datasets where only some of the instances have types, and/or new types should be introduced besides assigning instances to existing ones. The full paper has already been recently presented at the SSDBM [11] conference, whereas a video demonstrating our system is also available online¹.

2. System Architecture

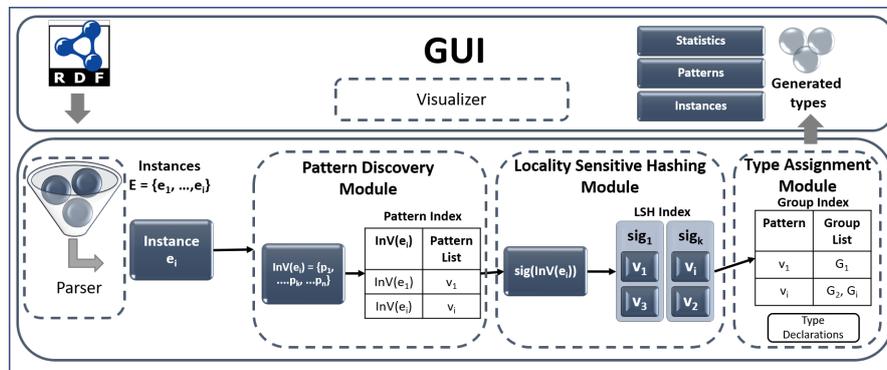


Figure 1: *HInT* high-level architecture.

The high-level architecture of our system is shown in Fig. 1. It consists of the graphical user interface (GUI) and the back-end subsystem. The GUI has been implemented in HTML and CSS

¹<https://users.ics.forth.gr/~kondylak/HInT.mp4>

using the Vue JS framework whereas the back-end is implemented in Python.

GUI. Using the GUI the users are able to upload an RDF dataset, for which they would like to discover the schema or to provide a URL for the file, if it is already available on the web. The file is then processed through the back-end subsystem and the identified types are presented along with statistics. In addition discovered patterns for each type are shown and example instances are presented. The bigger the types on the interface the more instances they contain. In addition, users are able to select the identified types in order to see the number of instances available for that type, to visualize example instances, and to see the patterns identified for that type. Furthermore, if the ground truth of the dataset is available it can be loaded as well and graphs for the precision, the recall and the f-measure are presented. A screenshot is shown in Fig. 2.

Back-end. Pattern Discovery Module. As soon as the RDF file is parsed, each instance is described using an *instance vector*. An instance vector contains the properties of the specific instance, as properties provide a descriptive representation of it. In the instance vector, we do not store potentially available typing information as types are later considered in the type assignment module. In our approach, we first try to reduce instance processing, to pattern processing. As such, based on the instance vectors of the various instances, we gradually identify the patterns existing in a dataset.

Note that multiple types can be assigned to an instance vector and as a consequence to a pattern. A pattern represents a set of instances that use exactly the same set of properties. As a result, patterns improve the efficiency of the approach by: (i) avoiding unnecessary processing on the subsequent index structures for instances having the same structure, e.g., insertions, queries, etc., and (ii) reducing the memory footprint by not storing duplicate information, e.g., storing multiple times the same instance vector for similar instances.

Locality Sensitive Hashing Module. To achieve a native incremental type discovery, we use a Locality-Sensitive Hashing (LSH) method. The major advantage of LSH is the fact that each input is treated independently. As a result, we can avoid the pairwise comparisons between the constructed patterns. In our approach, we build an *LSH index* according to the hash values provided for the patterns in our dataset. As described earlier, when a new instance arrives for which a pattern does not exist, a new pattern is generated. The LSH module is responsible for generating the signature based on the generated pattern, which is then used to insert the pattern to the LSH Index. Then, in order to retrieve similar patterns, we query the LSH index using a pattern's signature. When such a query is issued, a group of similar patterns will be returned, that are candidates for belonging to the same type - remember that LSH is an approximate algorithm. Note that LSH only works for retrieving similar instances and not for type assignment which is happening in a following step.

Type Assignment Module. The type assignment module builds *groups* of similar patterns according to the values of their signatures from the sensitive-hashing functions and according to the provided typing declarations. Each pattern can be assigned to one or more groups, whereas a group represents a collection of patterns. The main idea of the algorithm is that the classification of a pattern does not fully depend on the results retrieved by LSH. On the contrary, the available type declarations in the dataset are not only used to complement the results retrieved by LSH, but they also have a high impact on the classification process.

3. Demonstration Scenario & Conclusions

The goal for the demonstration is to show, through experimentation, the challenges involved in semantic schema discovery and to understand the benefits of a hybrid and incremental approach like *HInT*. We will start by explaining the objective of semantic schema discovery and guiding

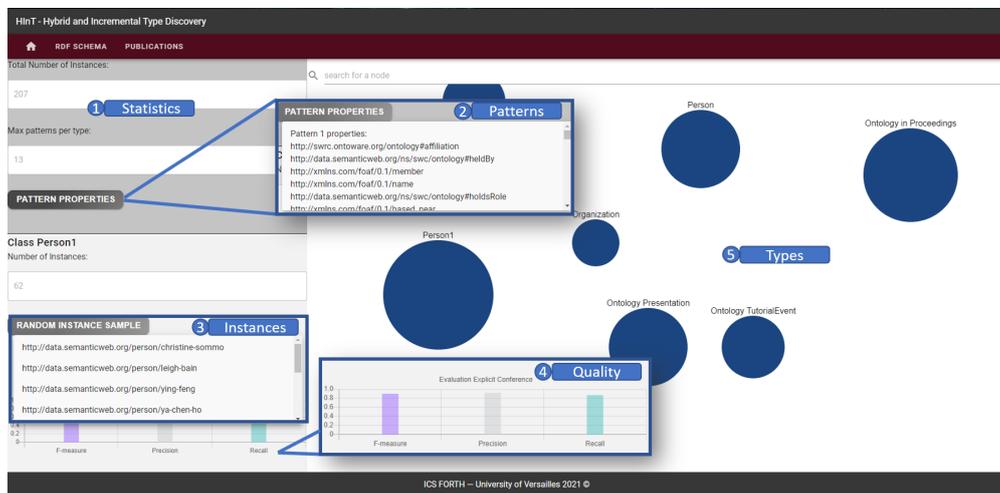


Figure 2: Screenshot from schema discovery using *HInT*.

participants through the different design choices for *HInT*. We will then walk them through the following steps.

Configuration. We will use five datasets which will be presented to the users: BNF, Conference, DBpedia, HistMunic and LUBM ranging in both size (from 381 to 91 million triples), and in complexity. LUBM is a synthetic benchmark and includes an ontology along with a data generator, which was used to generate 91M triples. The Conference dataset consists of 1,430 triples. The BNF dataset exposes data for the French National Library and contains 381 triples. The third dataset is extracted from DBpedia and contains 19,696 triples. Finally, histmunic is an open government dataset that contains 119,151 triples. For each dataset, we will show non-trivial examples, where schema information is missing and should be inferred.

Schema Discovery Ignoring Known Types. In this step, we will configure *HInT* to ignore typing information that might already be available in the dataset. Then we will explain the various steps for discovering the schema and the algorithms used for pattern discovery, LSH indexing and type assignment. Upon each execution the detected types will be presented along with statistical information (e.g. the number of instances and patterns for each type) and the user will be able to explore the patterns for a selected type and also visualize a sample of its instances. Further, as for these datasets we already know the ground truth, the precision, recall and f-measure will be graphically visualized and a discussion on the quality of the discovered types will be made.

Schema Discovery Exploiting Typing Information. In this step, we will run exactly the same scenarios as in the previous step, but this time we will configure the system to exploit available typing information. Again the various steps for pattern discovery, LSH indexing and

type assignment will be presented, explaining where the type information is used for improving the discovered types. The precision, recall and f-measure will be calculated for the various datasets and compared with the previous step, demonstrating the improvement of the schema discovery task.

Conclusions. We demonstrate *HInT*, the first hybrid and incremental type discovery system for RDF datasets. The core innovations are that a) we reduce instance comparisons to pattern comparisons, b) we employ LSH functions that enable incremental schema discovery and c) we can both detect missing types of available instances and generate new types that are missing. We demonstrate the versatile new performance and trade-offs that *HInT* provides, and we allow conference participants to understand the challenges of such a complex task. Finally, we allow conference participants to actively explore the developed infrastructure and the corresponding algorithms.

Acknowledgments

This research project was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the “2nd Call for H.F.R.I. Research Projects to support PostDoctoral Researchers” (iQARuS Project No 1147).

References

- [1] K. Kellou-Menouer, et al., A survey on semantic schema discovery, *VLDB Journal* (2021).
- [2] Y. Tsuboi, N. Suzuki, An algorithm for extracting shape expression schemas from graphs, in: *ACM DocEng*, 2019, pp. 32:1–32:4.
- [3] A. Lutov, S. Roshankish, M. Khayati, P. Cudré-Mauroux, Statix—statistical type inference on linked data, in: *IEEE Big Data*, 2018, pp. 2253–2262.
- [4] K. Christodoulou, N. W. Paton, A. A. Fernandes, Structure inference for linked data sources using clustering, in: *Trans. on Large-Scale Data-and Knowledge-Centered Systems XIX*, 2015, pp. 1–25.
- [5] H. Paulheim, Browsing linked open data with auto complete, *Semantic Web Challenge* (2012).
- [6] A. G. Nuzzolese, A. Gangemi, V. Presutti, P. Ciancarini, Type inference through the analysis of wikipedia links, in: *LDOW*, 2012.
- [7] H. Paulheim, C. Bizer, Type inference on noisy rdf data, in: *ISWC*, 2013, pp. 510–525.
- [8] L. Fang, Q. Miao, Y. Meng, Dbpedia entity type inference using categories, in: *ISWC 2016 Posters & Demos*, 2016.
- [9] H. Kondylakis, D. Plexousakis, Ontology evolution in data integration: Query rewriting to the rescue, in: *ER*, volume 6998, Springer, 2011, pp. 393–401.
- [10] H. Kondylakis, D. Plexousakis, Ontology evolution: Assisting query migration, in: *ER*, volume 7532, Springer, 2012, pp. 331–344.
- [11] N. Kardoulakis, et al., Hint: Hybrid and incremental type discovery for large RDF data sources, in: *SSDBM*, 2021.