Linkex: A Tool for Link Key Discovery Based on Pattern Structure

Nacira Abbas*, Jérôme David**
Amedeo Napoli*

*Université de Lorraine, CNRS, Inria, Loria, F-54000 Nancy, France,
nacira.abbas@inria.fr
amedeo.napoli@loria.fr
**Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
jerome.david@inria.fr

Abstract. Links constitute the core of Linked Data philosophy. With the high growth of data published in the web, many frameworks have been proposed to deal with the link discovery problem, and particularly the identity links. Finding such kinds of links between different RDF data sets is a critical task. In this position paper, we focus on link key which consists of sets of pairs of properties identifying the same entities across heterogeneous datasets. We also propose to formalize the problem of link key discovery using Pattern Structures (PS), the generalization of Formal Concept Analysis dealing with non binary datasets. After providing the proper definitions of link keys and setting the problem in terms of PS, we show that the intents of the pattern concepts correspond to link keys and their extents to sets of identity links generated by their intents. Finally, we discuss an implementation of this framework and we show the applicability and the scalability of the proposed method.

1 Introduction

Data interlinking is the task of finding the same entities described in different RDF datasets. Two main approaches have been proposed to perform this task: numerical-based methods that measure a similarity between entities and consider that the closest the entities, the more likely they are the same (Volz et al., 2009; Ngomo and Auer, 2011). Logical-based approaches where logical rules are used, which express sufficient conditions for two entities to be the same (Saïs et al., 2007; Al-Bakri et al., 2015, 2016; Hogan et al., 2012). One of these approaches is based on *link keys* (Atencia et al., 2014a) that extend the notion of a key used in databases. Link keys are rules allowing to infer identity links between RDF datasets. In practice, a link key is a set of pairs of properties from two datasets that generate `same-as` links.

An example of a link key is:

$$\{\langle \mathsf{auteur}, \mathsf{creator}\rangle\}\{\langle \mathsf{titre}, \mathsf{title}\rangle\}\ \textit{linkkey}\ \langle \mathsf{Livre}, \mathsf{Book}\rangle$$

stating that whenever an instance of the class Livre has the same values for property auteur as an instance of class Book has for property creator and they share at least one value for their property titre and title, then they denote the same entity. An algorithm to extract link key candidates is proposed in (Atencia et al., 2014a). However, these candidates are not yet link keys. In order to select the best link key candidate to apply as a link key, supervised and unsupervised measures have been proposed to asses the quality of link key candidates but this is out of the scope of this paper.

The question of using Formal Concept Analysis (FCA) to extract such kind of keys has naturally arisen, given that the set of link key candidates is an ordered structure. A first formalization is given in (Atencia et al., 2014b) then developed in (Atencia et al., 2019). Here, given two classes belonging to different RDF datasets, to find link key candidates among these classes, an encoding of formal context is proposed where a relation is defined between a set of objects, which correspond to pairs of instances, and a set of scaled attributes obtained by applying existential and universal scaling operators to the set of pairs of properties. Scaling (binarizing) data may not be efficient, particularly when we have to deal with a large number of entries like RDF datasets. In this position paper, instead of scaling, we propose to work directly on link key expressions which are the syntactic formulations of link keys. To do that, we use Pattern Structures (Ganter and Kuznetsov, 2001), a generalization of FCA to deal with non binary data, where objects are pairs of instances and their descriptions are link key expressions. We show that the intents of resulting pattern concepts correspond to link key candidates and their extents to sets of identity links generated by their intents.

In the following, we first give some definitions and notations. Then, we introduce a Pattern Structure view of link key extraction problem. Finally, we present a prototype tool which extracts link key candidates using Pattern Structures and we show the applicability and scalability of this tool by applying it to real datasets.

2 Notations and definitions

An RDF dataset is a set of triples:

$$\langle subject, property, object\rangle \in (U \cup B) \times U \times (U \cup B \cup L)$$

Where U is a set of IRIs (Internationalized Resource Identifier), B a set of blank nodes i.e. anonymous resources and L a set of literals, i.e., values depending on datatypes. In this work, we consider only objects which are literals. To avoid confusion with Pattern Structures objects, we refer to "object" in an RDF triple as "rdf object".

Let $D, D^{'}$ be two RDF datasets, we aim to discover link keys between two specific classes $C, C^{'}$ from $D, D^{'}$ respectively.
$C = \{s | \langle s, \mathit{rdf{:}type}, C\rangle \in D\}$ and $C^{'} = \{s^{'} | \langle s^{'}, \mathit{rdf{:}type}, C^{'}\rangle \in D^{'}\}$ are the sets of instances C and $C^{'}$ respectively. $P = \{p | \exists s, o,\ \langle s, p, o\rangle \in D\}$ and $P^{'} = \{p^{'} | \exists s^{'}, o^{'},\ \langle s^{'}, p^{'}, o^{'}\rangle \in D^{'}\}$ are the sets of properties from D and $D^{'}$ respectively.

Unlike databases, the properties in RDF are not functional i.e. for one property, an instance may have more than one value i.e. rdf object.
Here, $p(s) = \{o | \exists s, p,\ \langle s, p, o\rangle \in D\}$ is the set of the values of an instance s for the property p.

Figure 1 represents an example of two datasets. The first one, contains instances of the class *Person* and the second one contains instances of the class *Inhabitant*. An example of an RDF triple here is $\langle i_1, name, Dubois\rangle$, expressing that i_1 has the value "Dubois" for the property *name*. The goal here is to find a link key which identifies the same entities described in both

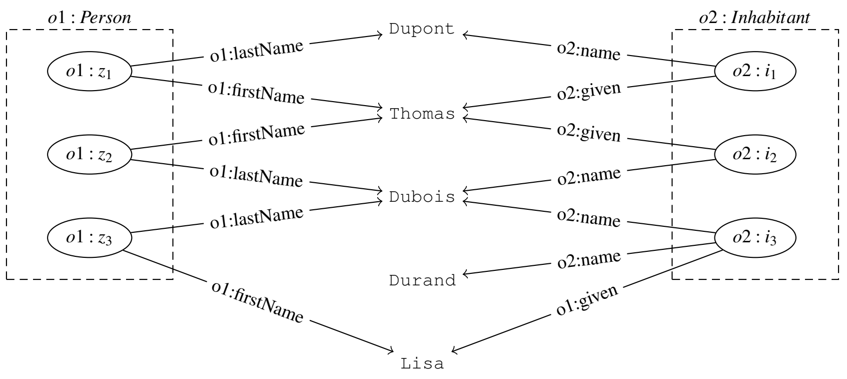


FIG. 1 – *Example of two datasets representing respectively instances of classes Person and Inhabitant.*

classes. One may expect to identify that z_1 is the same entity as i_1 i.e. the *link* $\langle z_1, i_1\rangle$ as well as the links $\langle z_2, i_2\rangle$ and $\langle z_3, i_3\rangle$.

A link key is a statement of the form:

$$\{\langle p_1, p_1^{'}\rangle, .., \langle p_k, p_k^{'}\rangle\}\{\langle p_1, p_1^{'}\rangle, .., \langle p_k, p_k^{'}\rangle, .., \langle p_n, p_n^{'}\rangle\} \ linkkey \ \langle C, C^{'}\rangle$$

meaning that for all pairs of instances $\langle s, s^{'}\rangle \in C \times C^{'}$, if s and $s^{'}$ have the same values, i.e. rdf objects, for each pairs of properties $p_{1,..,k}$ and $p^{'}_{1,..,k}$ respectively, and share at least one value for each pair of properties $p_{k+1,..,n}$ and $\mathrm{p}^{'}_{k+1,..,n}$ respectively, then they represent the same entity i.e. $(\langle s, \texttt{owl:sameAs}, s^{'}\rangle)$.

Definition 1 (Link key expression) *Let Eq and In be sets of pairs of properties $\langle p, p^{'}\rangle \in P \times P^{'}$ where $Eq \subseteq In$. $K = \langle Eq, In, \langle C, C^{'}\rangle\rangle$ is called a link key expression over two classes $C, C^{'}$. Notice that Eq and In may be empty.*

Since $Eq \subseteq In$, we write $K = Eq, In \setminus Eq$ for short. Which means that when a pair of properties belongs to the Eq part, we don't represent it in the In part.

As example of link key expression over the two classes *Person* and *Inhabitant*, we have: $\{\}, \{\langle lastName, name\rangle, \langle firstName, given\rangle\}$ where Eq is empty.

Definition 2 (Satisfaction of a link key expression) *We say that a link $\langle s, s^{'}\rangle \in C \times C^{'}$ satisfies the link key expression $K = Eq, In$ and we write $K \models \langle s, s^{'}\rangle$ iff:*

$$\forall \langle p, p^{'}\rangle \in Eq \Longrightarrow p(s) = p^{'}(s^{'}) \neq \emptyset \ and \ \forall \langle p, p^{'}\rangle \in In \Longrightarrow p(s) \cap p^{'}(s^{'}) \neq \emptyset$$

Notice that a link may satisfies more than one link key expression.

The set $L(K) = \{\langle s, s^{'}\rangle \in C \times C^{'} | K \models \langle s, s^{'}\rangle\}$ is the set of all links satisfying K and called the *link set* generated by K. In the above example, the link $\langle z_3, i_3\rangle$ satisfies the link key expression $\{\langle firstName, given\rangle\}, \{\langle lastName, name\rangle\}$, but it doesn't satisfy the link key expression: $\{\langle firstName, given\rangle, \langle lastName, name\rangle\}, \{\}$ since the instances z_3 and i_3 don't have the same values for the properties *lastName* and *name* respectively.

Definition 3 (meet, join, subsumption of link key expressions) *Let $K_1 = Eq_1, In_1$ and $K_2 = Eq_2, In_2$ two link key expressions. The meet $\sqcap$ and join $\sqcup$ of K_1 and K_2 are defined respectively as follow:*

$$K_1 \sqcap K_2 = Eq_1 \cap Eq_2, In_1 \cap In_2 \text{ and } K_1 \sqcup K_2 = Eq_1 \cup Eq_2, In_1 \cup In_2$$

We say K_2 subsumes K_1 and we write $K_1 \sqsubseteq K_2$ iff $K_1 \sqcap K_2 = K_1$ where $\sqsubseteq$ is a partial order between link key expressions.

For example we have:
$K_1 = \{\langle lastName, name\rangle, \langle firstName, given\rangle\}, \{\}$
$K2 = \{\langle firstName, given\rangle\}, \{\}$
$K_1 \sqcap K_2 = \{\langle firstName, given\rangle\}, \{\} = K_2$ which means that $K_2 \sqsubseteq K_1$.
$K_1 \sqcup K_2 = \{\langle lastName, name\rangle, \langle firstName, given\rangle\}, \{\}$.

3 Link key extraction with Pattern Structure

Pattern Structures (Ganter and Kuznetsov, 2001) are a generalization of FCA (Ganter and Wille, 2012) to deal with complex data such as graphs or intervals. While FCA starts from a binary relation between objects and attributes, Pattern Structures relates each object to its description. In follows, we define a Pattern Structure for link key extraction.

Definition 4 (Pattern Structure for link key extraction) *A Pattern Structure for link keys extraction is a triple $(C \times C^{'}, (D, \sqcap, \sqcup), \delta)$ where, $C \times C^{'}$ is the set a pair of instances $\langle s, s^{'}\rangle \in C \times C^{'}$. D is a set of link key expressions over the two classes C and $C^{'}$. δ is a mapping associating each pair of instances $\langle s, s^{'}\rangle \in C \times C^{'}$ to its description K, defined as the maximal link key expression satisfied by $\langle s, s^{'}\rangle$ i.e. the join of the link expressions satisfied by $\langle s, s^{'}\rangle$. Formally,*

$$\delta(\langle s, s^{'}\rangle) = \sqcup\{K | K \models \langle s, s^{'}\rangle\}$$

$(D, \sqcap, \sqcup)$ is a lattice. The *meet*, *join* and subsumption of two link key expressions are defined as in Definition 3.

From the previous example of datasets represented in Figure 1, we obtained the Pattern Structure in Table 1. Here for example, to find $\delta(\langle z_1, i_1\rangle)$, we have computed the join of all link key expressions satisfied by the link $\langle z_1, i_1\rangle$ and we obtained the description $\delta(\langle z_1, i_1\rangle) = \{\langle lastName, name\rangle, \langle firstName, given\rangle\}, \{\}$.

Notice that we omit pairs of instances for which descriptions correspond to the empty link key expression $K = \{\}, \{\}$.

Derivation operators $.^{\square}$ are defined as follows for $A \subseteq C \times C^{'}$ and $K \in D$:

$$A^{\square} = \sqcap_{\langle s, s^{'}\rangle \in A} \delta(\langle s, s^{'}\rangle) \qquad \text{and} \qquad K^{\square} = \{\langle s, s^{'}\rangle \, | K \sqsubseteq \delta(\langle s, s^{'}\rangle)\}$$

(A, K) is a *pattern concept* if $A^{\square} = K$ and $K^{\square} = A$. We say that a pattern concept (A, K) is subsumed by another pattern concept (A_1, K_1) if $A \subseteq A_1$ (or equivalently if $K_1 \sqsubseteq K$). From the lattice represented in Figure 2 we have the pattern concept
$\{\langle z_1, i_1\rangle, \langle z_2, i_2\rangle\}$, $\{\langle lastName, name\rangle, \langle firstName, given\rangle\}, \{\}$.

We aim to extract link key candidates using the Pattern Structure defined above. First we give a formal definition of a link key candidate :

Definition 5 (Link key candidate) *A link key expression K is a link key candidate over two classes C, C' if $L(K) \neq \emptyset$ and there is no link key expression H such as $H \sqsubseteq K$ that generates the same link set as K i.e. $L(K) \neq L(H)$*

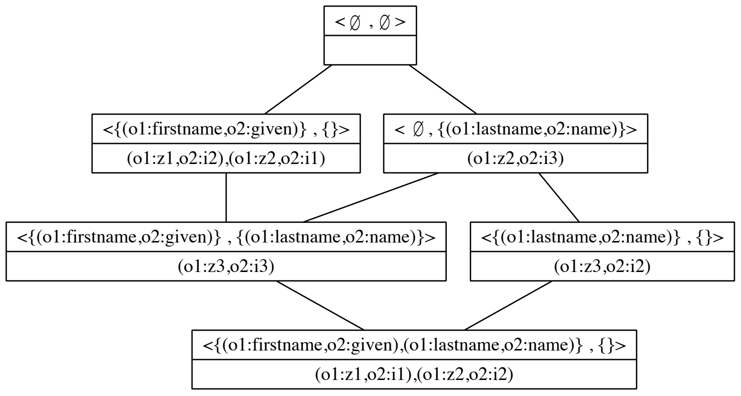
For example, the link key expression $\{\langle lastName, name\rangle, \langle firstName, given\rangle\}, \{\}$ is a link key candidate. Where as $\{\langle lastName, given\rangle\}, \{\}$ is not, because it doesn't generate any link.

One can see that, if (A, K) is a pattern concept, then its intent K represents a link key candidate and its extent is the link set generated by this link key candidate.

For example, the pattern concept:
$\{\langle z_1, i_1\rangle, \langle z_2, i_2\rangle, \langle z_3, i_3\rangle\}$, $\{\langle firstName, given\rangle\}, \{\langle lastName, name\rangle\}$, identify the link key candidate $\{\langle firstName, given\rangle\}, \{\langle lastName, name\rangle\}$, i.e. the intent, which generates the link set $\{\langle z_1, i_1\rangle, \langle z_2, i_2\rangle, \langle z_3, i_3\rangle\}$ i.e. the extent. This link key candidate can be applied as a link key. In fact, it generates the expected links. This link key states that whenever an instance of the class *Person* has same values for property $firstName$ as an instance of class $Inhabitant$ has for property *given* and they share at least one value for their property $lastName$ and *name* then they denote the same entity. For example the instances z_3 and i_3 represent the same person since both of them had the same first name and share at least one family name. The class $Inhabitant$ gives, for example, the birth name and the married name while the class *Person* gives just one of these two.

	Eq	*In*
$\langle z_1, i_1\rangle$	$\{\langle lastName, name\rangle, \langle firstName, given\rangle\}$	$\{\}$
$\langle z_1, i_2\rangle$	$\{\langle firstName, given\rangle\}$	$\{\}$
$\langle z_2, i_1\rangle$	$\{\langle firstName, given\rangle\}$	$\{\}$
$\langle z_2, i_2\rangle$	$\{\langle lastName, name\rangle, \langle firstName, given\rangle\}$	$\{\}$
$\langle z_2, i_3\rangle$	$\emptyset$	$\{\langle lastName, name\rangle\}$
$\langle z_3, i_2\rangle$	$\{\langle lastName, name\rangle\}$	$\{\}$
$\langle z_3, i_3\rangle$	$\{\langle firstName, given\rangle\}$	$\{\langle lastName, name\rangle\}$

TAB. 1 – *The pattern structure computed from RDF datasets represented in Figure 1*



< ∅ , ∅ >
<{(o1:firstname,o2:given)} , {}>
(o1:z1,o2:i2),(o1:z2,o2:i1)
< ∅ , {(o1:lastname,o2:name)}>
(o1:z2,o2:i3)
<{(o1:firstname,o2:given)} , {(o1:lastname,o2:name)}>
(o1:z3,o2:i3)
<{(o1:lastname,o2:name)} , {}>
(o1:z3,o2:i2)
<{(o1:firstname,o2:given),(o1:lastname,o2:name)} , {}>
(o1:z1,o2:i1),(o1:z2,o2:i2)

FIG. 2 – *The resulting lattice from the pattern structure in Table 1.*

The resulting pattern concept lattice presents in an elegant way the space of link key candidates and the links they generate. When the resulting lattice is small, an expert has the possibility to navigate the lattice in order to choose an appropriate link key. In the opposite case i.e. when the lattice is big, one may apply quality measures to rank the resulting link key candidates and select the best one to apply. These two options are available in the impleman-tation that we present in this paper.

4 Tool Description

We have implemented a prototype tool called `Linkex` [1], which aims to extract link key candidates using Pattern Structures. It starts from two RDF data sets and generates a pattern concept lattice. The intent of each resulting pattern concept corresponds to a link key candidate and its extent to the set of identity links generated by its underlying intent (link key candidate). In practice, the two datasets are given as files (which may be encoded in RDF/XML, turtle or n-triples) or retrieved in web datasets thanks to SPARQL constraint query. `Linkex` performs the following treatments:

1. **From RDF datasets to pattern structure**: First we proceed to value normalization: putting all alphabetic characters in lowercase, tokenization, removal of diacritics and spaces and finally sorts token sequences in alphabetical order. Then we remove non-discriminating properties (i.e., properties for which all instances have the same value) and properties with weak support (i.e., which have very little frequency). Finally, we compute the description of each pair of instances and construct the Pattern Structure.
2. **Building the pattern concept lattice**: We generate the pattern concept lattice using a modified version of `AddIntent` algorithm (Van Der Merwe et al., 2004) .
3. When the resulting pattern concept lattice is small, `Linkex` gives the possibility to visualize link key candidates in a nice way i.e. organized in a lattice represented by a Hasse diagram. For example the Figure 2, which represents a pattern concept lattice, is a direct output of this tool.
4. Since, the resulting pattern concept lattice doesn't tell us which link key candidate to select as a link key. This tool offer the possibility to asses the quality of the extracted link keys candidates using supervised (recall, precision) and unsupervised measures (coverage, discriminability). See (Atencia et al., 2014a) for details.

To demonstrate the applicability and the scalability of the proposed tool, we have experi-mented with two bibliographical datasets, that contains description of authors and the books that they have written. The first one [2], called here BnF, is produced by BnF, the "Bibliothèque nationale de France" which is the French national library. The second one [3], called here IdRef, is produced by ABES, "Agence Bibliographique de l'Enseignement Supérieur" which is the french bibliographic agency of academic libraries.

We aim here to find link key candidates between instances of the classes representing au-thors contained in both datasets. However, the datasets are large, for example the BnF dataset contains 2,221,471 authors. For this reason, BnF and ABES, have provided us with two sam-ples of data for the experiments.

1. https://gitlab.inria.fr/moex/linkex
2. https://data.bnf.fr
3. https://www.idref.fr

The first sample, called variant 1, is an extraction of authors instances (and their books instances) that have a combination firstname, name which is in the top 1000 most frequent homonyms. The second sample, called variant 2, is an extraction of all authors instances (and their book instances) that have a name starting with letter 'A'. We have performed these experiments using a MacBookPro11,3, Intel Core i7 @2,3 GHz with 10GB RAM allocated to the Java virtual machine.

As showed in Table 2, we got short running times comparing to the size of processed data. Notice that, the processing time is correlated to the size of datasets. Yet, the most interesting result here is the number of the obtained pattern concepts i.e. the number of link key candidates represented by the column **#PatternConcepts** in Table 2. For example, in variant 1, from 1,564,495 pattern structure descriptions, we got only 155 link key candidates.

As a direct consequence of this, is that an expert could easily navigate these link key candidates and evaluate them. Thus, we can say that the proposed method implemented as Linkex is applicable and scalable.

Variant	**#BnF**	**#IdRef**	**#PsEntries**	**#PatternConcepts**	**Processing time**
Variant 1	15,421	8,162	1,564,495	155	2m10
Variant 2	142,571	18,637	12,348,012	186	6m50

TAB. 2 – *Experimentation results*

5 Conclusions

In this work, we formalize the problem of extracting link key candidates using Pattern Structures and we propose a tool allowing to automatically build a pattern concept lattice where each pattern concept intent is a candidate link key. We aim to extend this work to consider interdependent classes i.e when rdf objects are instances of other classes and we plan also to relax equality constraints used to compare literals by considering similarity measures instead.

Acknowledgments

This work has been supported by the ANR project Elker (ANR-17-CE23-0007-01) and the Bibliothèque Nationale de France in the context of the agreement between Inria and Ministère de la culture.

References

Al-Bakri, M., M. Atencia, J. David, S. Lalande, and M.-C. Rousset (2016). Uncertainty-sensitive reasoning for inferring same as facts in linked data. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pp. 698–706. IOS press.

Al-Bakri, M., M. Atencia, S. Lalande, and M.-C. Rousset (2015). Inferring same-as facts from linked data: an iterative import-by-query approach. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Atencia, M., J. David, and J. Euzenat (2014a). Data interlinking through robust linkkey extraction. In *ECAI*, pp. 15–20.

Atencia, M., J. David, and J. Euzenat (2014b). What can fca do for database linkkey extraction? In *3rd ECAI workshop on What can FCA do for Artificial Intelligence?(FCA4AI)*, pp. 85–92. No commercial editor.

Atencia, M., J. David, J. Euzenat, A. Napoli, and J. Vizzini (2019). Link key candidate extraction with relational concept analysis. *Discrete Applied Mathematics*.

Ganter, B. and S. O. Kuznetsov (2001). Pattern structures and their projections. In *ICCS*, pp. 129–142. Springer.

Ganter, B. and R. Wille (2012). *Formal concept analysis: mathematical foundations*. Springer Science & Business Media.

Hogan, A., A. Zimmermann, J. Umbrich, A. Polleres, and S. Decker (2012). Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *Web Semantics: Science, Services and Agents on the World Wide Web 10*, 76–110.

Ngomo, A.-C. N. and S. Auer (2011). Limes - a time-efficient approach for large-scale link discovery on the web of data. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Saïs, F., N. Pernelle, and M.-C. Rousset (2007). L2r: A logical method for reference reconciliation. In *Proc. AAAI*, pp. 329–334.

Van Der Merwe, D., S. Obiedkov, and D. Kourie (2004). Addintent: A new incremental algorithm for constructing concept lattices. In *ICFCA*, pp. 372–385. Springer.

Volz, J., C. Bizer, M. Gaedke, and G. Kobilarov (2009). Silk-a link discovery framework for the web of data. *LDOW 538*.

Résumé

Links constitute the core of Linked Data philosophy. With the high growth of data published in the web, many frameworks have been proposed to deal with the link discovery problem, and particularly the identity links. Finding such kinds of links between different RDF data sets is a critical task. In this position paper, we focus on link key which consists of sets of pairs of properties identifying the same entities across heterogeneous datasets. We also propose to formalize the problem of link key discovery using Pattern Structures (PS), the generalization of Formal Concept Analysis dealing with non binary datasets. After providing the proper definitions of link keys and setting the problem in terms of PS, we show that the intents of the pattern concepts correspond to link keys and their extents to sets of identity links generated by their intents. Finally, we discuss an implementation of this framework and we show the applicability and the scalability of the proposed method.