PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY CENTER OF NAAMA

DEPARTMENT: MATHE AND COMPUTING

SPECIALTY: Information Systems

Knowledge Graph Consistency

- Presented with:

ABDELALI Mohammed Belkaid

Bouzidane Ahmed

- Framed by:

Atig Yahia

The academic year:

2019/2020

Dedication

This modest work is dedicated to:

To the one who has always been the source of

great affection and taste

of life ..... my MOTHER

To the one who has always been the source of

registration, courage has everything

throughout my studies ..... my FATHER

To all my brothers and sisters

To all my family and friends.

To all my friends are exceptions.

To my colleague AHMED

To all the " Information Systems " 2019/2020

promotion that I wish them a good future.

To all those who have helped me from near or

far to achieve this

modest work

Belkaid

DEDICATION

This modest work is dedicated to:

To the one who has always been the source of

great affection and taste

of life ..... my MOTHER

To the one who has always been the source of

registration, courage has everything

throughout my studies ..... my FATHER

To all my brothers and sisters

To all my family and friends.

To all my friends are exceptions.

To my colleague BELKAID

To all the " Information Systems" 2019/2020

promotion that I wish them a good future.

To all those who have helped me from near or

far to achieve this

modest work

AHMED

ACKNOWLEDGMENT

First of all, we would like to thank Allah,
the merciful and the merciful to have us
gave the strength and courage to carry out this
job.
We would like to express our spirits
thanks to our supervisor “MR Atig
Yahia ”, for guidance and advice
that he was introducing us to this wonderful field
and able to provide us during the evolution of
our project.
We would also like to thank all
teachers who have contributed to our
training.
Our thanks also go to all
and those who near or far have us
brought help and encouragement,

Contents

List of Figures

List of Tables

List of Algorithms

e

e

Abstract

One of the current problems of knowledge graphs is the existence of erroneous assertion, what affect the consistency and subsequently usability of the graph. The aim of this modest work is to fix that issue to get a consistent graph by exploit a framework serves this purpose.

Keywords: Knowledge Graphs, Assertion Correction, Constraint Mining, Consistency Checking.

Chapter 1: Introduction

Knowledge graph become a popular knowledge representation nowadays like Wikidata and Dbpedia due to its usefulness and strength. There are so many areas touch it such as: search engines, question answering, common sense reasoning, data integration and machine learning or any artificial intelligence agent; without talking about private use or commercial exploitation. The problem is that these areas are suffering from quality issues that limit the usefulness of the graphs. At the end the data provided valuable only if it is accurate.

One of the methods proposed in the literature to improve the quality of KGs (Knowledge Graphs) is to ensure their consistency by correcting erroneous assertions. In this context, we address the issue of the consistency of KGs and we make the following contributions:

1 - Present the concept of data models and choose those data models that have most impact in nowadays to talk about.

2 - Trying to define the concept of the knowledge graph, when we track the term deep in our way to understand the term that have a lot of clamor around -where we collect to many different definition, and take it from the first use of term of graph itself -it and give our point of view about it to end with defining the types of KG.

3 - Differentiation between the term of the data graph and KG where's explain the layers that transform the former to the last.

4 - How can the KG can be (semi-)automatic with detective and inductive methods.

5 - Give an overview about the exist KGs.

6 - Give a look for KG quality assessment.

7 - Define the inconsistency/consistency with the mention of different point of view about the inconsistency concept.

8 - Talk about how can detect the inconsistency and the ways to handle it and it causes.

9 - Present last work to correct erroneous assertion according to our research with explanation to refine the consistency of KGs, where the framework also uses the latest technique for deep learning that are for now that are the art-state for now.

The thesis is organized as follows[1]: in the second chapter, we present the basic knowledge to understand the subject. In the third chapter, we examine the consistency concept and present a framework to refine inconsistency for KG by correcting the erroneous assertion. Finally, we conclude in the last chapter with a discussion of the obtained results and a set of perspectives for the near future.

[1] We content ourselves by the fundamental concepts for the framework so as not to widen the chapters further, and when a section or part contains many pieces of information that not really for specific resource, we refer all the resource together, but when the information it is for specific resource in that section or part we refer right after it.

Chapter 2: Basic knowledge

1 Introduction

In this chapter we will clarify the concept and knowledge that are essential to understand the rest of the thesis, where: in the first section, we present general view about some important data models to know the role of graph model exactly and his position from other models, while the second section talks about the concept of knowledge graph in deep because of the obscurity about that term, than at the third section we go more deep to differentiate between graph data model to know and knowledge graph, and in the fourth section we talked about the semi/ automation in the KG where we talked about deductive and inductive reasoning briefly -that the realization of consistency should not be manually- in the area we need because that topic is very large, next we look for some KGs that are in use, and finally conclude the chapter with a common graph assessment quality.

2 Data models

A data model is a collection of conceptual tools for describing the real-world entities to be modeled in the database and the relationships among these entities.

The ideal data model provides:

1 - set of data structure types;

2 - set of operators or inference rules;

3 - set of integrity rules.

When look for a project to choose the right data model we take on consideration his identity to choose between transactions ACID or scalability by using BASE[1].

The good chose makes programming simpler and cheaper, respectively expensive -sometimes prohibitively expensive- to fix for the bad chose.

Take on consideration the combine of data models is possible.

[1] Basically Available, Soft state, Eventual consistency

Here over view for important models (the fig. 1 give a general of data models exist until 2008):

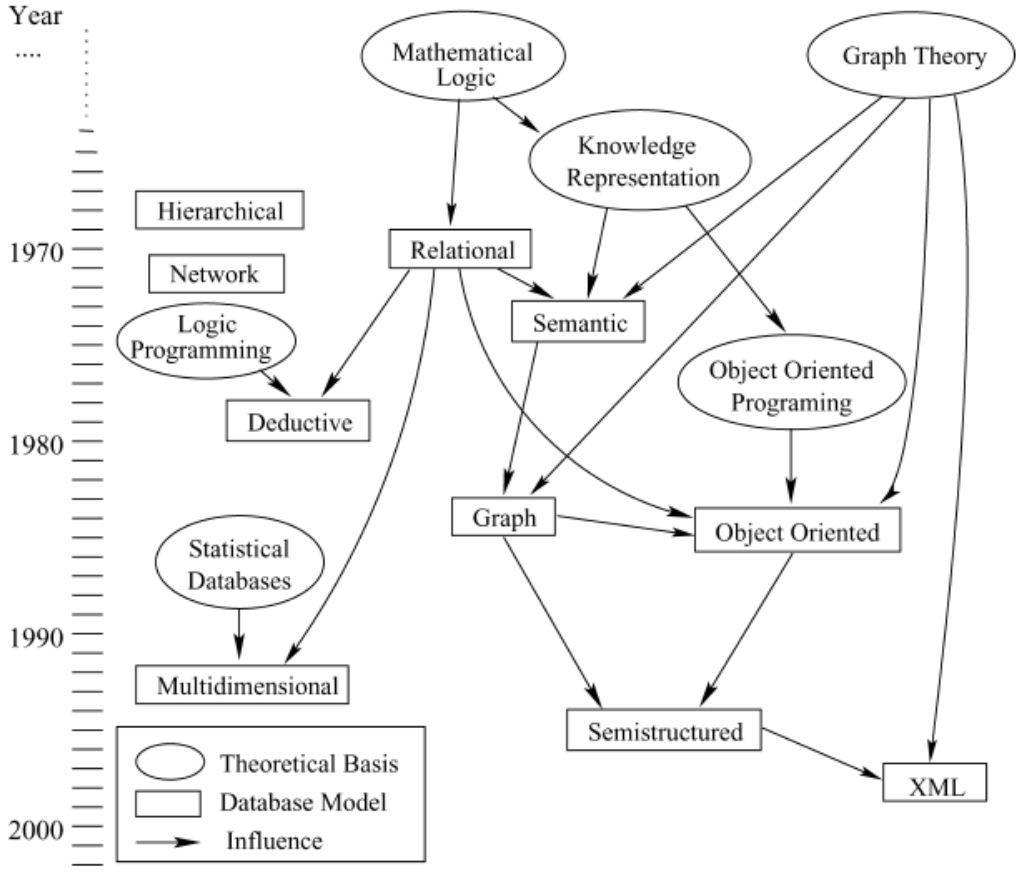

fig. 1.Evolution of database models (2008)[1]

2.1 The relational models

Introduced by Codd 1969.

The idea is store information with columns(category of information), rows(value for that category), to compose tables (can store data only for one object, person or product not both); for each tables there primary key that identifies the data in the table, and we can join the tables, the query language used it is SQL.

The relational DB based on the simple notion of relation, associated with algebra and logic. It remains useful (number one model used, according to fig. 2), when come to need:

- Stable and clean highly structured data (if your project requirements do not entail constant changes), also absence of duplication, and data integrity, establish the right connections

between all the tables, and maintain the structure intact; that is the main strengths of relational databases.

- Simple and uniform structure (when project have simple structure).

- Strong data accuracy: there's no repetitive or unstructured information (developers carefully sort through every piece of data), all records are correctly processed and stored.

Relational model falls apart when come to massive amounts of data and scale (the scaling it mean integration of new resources which mean the need to change the schema[1]), with the need for complex traversal queries with insanely speed querying or analytical capability, as well as more the grow of the web with his less structured nature in addition to the above with the need to speed, so it not recommended use for: search, recommendations, real-time nature application (in the input stage), complex data structure (because of no support for complex data types).

If we want to brevity the use case in a sentence it will be the answers for, what answers do I have? [1-5] [6]

359 systems in ranking, August 2020

Rank			DBMS	Database Model	Score		
Aug 2020	Jul 2020	Aug 2019			Aug 2020	Jul 2020	Aug 2019
1.	1.	1.	Oracle	Relational, Multi-model	1355.16	+14.90	+15.68
2.	2.	2.	MySQL	Relational, Multi-model	1261.57	-6.93	+7.89
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1075.87	+16.15	-17.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	536.77	+9.76	+55.43
5.	5.	5.	MongoDB	Document, Multi-model	443.56	+0.08	+38.99
6.	6.	6.	IBM Db2	Relational, Multi-model	162.45	-0.72	-10.50
7.	↑ 8.	↑ 8.	Redis	Key-value, Multi-model	152.87	+2.83	+8.79
8.	↓ 7.	↓ 7.	Elasticsearch	Search engine, Multi-model	152.32	+0.73	+3.23
9.	9.	↑ 11.	SQLite	Relational	126.82	-0.64	+4.10
10.	↑ 11.	↓ 9.	Microsoft Access	Relational	119.86	+3.32	-15.47
11.	↓ 10.	↓ 10.	Cassandra	Wide column	119.84	-1.25	-5.37

fig. 2.DB-Engines Ranking (August 2020,db-engines.com).

2.2 NoSQL

NoSQL is not a Standard, but most likely a movement[3].

First appeared of the term "NoSQL" in 1998, the meaning of this term change form No SQL to Not Only SQL over time, this term refers non-relational database technologies (models).

[1] Each such change requires a costly remodeling, reloading, and reindexing of data.

The business in those days look for high scalability, increased velocity, improved analytics, and social interaction and the growth of web with his distributed and multi type nature where the NoSQL technologies provide more value than relational, but for this feature NoSQL need to give up some ACID transactions characteristic (the KG it excepted [7] especially label Property Graph Model) and use BASE instead[1].

There many models under NoSQL, we choose four big influential database Key-Value, data graph, Document store, and wide-column stores data databases (or Tabular databases).[3]

If we want to brevity the use case in a sentence it will be the answers for, what questions do I have?

Key-Value:

Key value data store give up the ACID norms and use BASE to provide high speed (read and write as simple disk access) and highly partitionable and huge horizontal scaling that other database can't match for large-scaling, in other word give short time respond large-scale but may give stale (in case high frequent changeable data) data for users.[8, 9]

The mechanism is to stores values (the actual data) indexed by a key (the unique identifier) key-value pairs, but those values it opaque (can't process the data itself what means If the key is not known, the data can't be found and there no query language), and when the data be to huge (from key value DB view) the process of generating unique keys will be complex. other down side is the developer need to cover capabilities that we lose in coding for the higher speed. [9]

Document oriented store:

Extends the key value model and values (referred as “documents”) are stored in a typically in format that can be handled and processed by the machine and do some queries, typical format RDF, XML or JSON ... etc., documents and schemaless (what give it flexibility and nestibility).[10]

The flexible, semi structured, and hierarchical nature of documents and document databases allow to be a well adapt with applications needs in particular that dependent on efficient and effective for storing catalog information

[1] the developer chooses right model.

the terminology is not perfectly standardized, In the field of work recognize them as CRUD[1].

Graph Data model:

Widely used in networking structure problems solving(e.g. social data, biological, GPS or transport networks …etc.) and high-end analytics[2][3, 11]; because this type of knowledge abstraction offer concise and intuitive abstraction with graph nature Structures (Because of similarity between graph and object diagrams -Connected things-) it provide more value over relational model or his NoSQL alternatives.

Some characteristic guideline for use case:

- flexibility, as the schema (see section schema) can be defined later on project - Some call it schema-last-, what provides extensibility toward incomplete knowledge and capturing it without huge effort and data problems.

Some other data models like trees (XML ...) provide similar flexibility but require hierarchically data organizing unlike KG and that allow to represented cycles like we see fig. 4 [12].

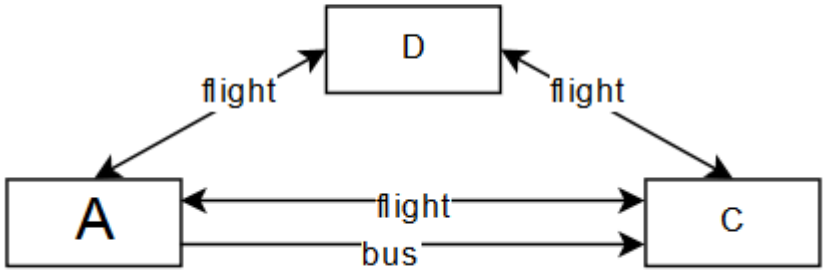

fig. 3 directed cycle in the routes between A, C, and D with flight.

- The only model even in NoSQL family that store relationship like relationships explicitly[13], that give significant benefits in terms of querying - especially the third generation[3] of graphs, potentially recursively finding entities connected through arbitrary-length paths -Graph databases are naturally indexed by relationships with other word element contains a direct pointer to its -.

[1] Creation/insertion, Retrieval (or query, search, read or find), Update, Deletion
[2] High-end analysis will often use advanced analysis techniques in practice (generally, machine learning (ML)by feed the models and improve their accuracy with generates new features based on the relationship analysis, and artificial intelligence (AI)).
[3] When the graph became independent on its own and had its own management systems, and not juste for low level.

Partner and Vukotic do performance experiment between relational database and a Neo4j graph database, by using 1,000,000 people each with approximately 50 friends, in basic social network to find friends-of-friends connections to a depth of five degrees.[13] full experiment [14].

Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

fig. 4. A performance experiment run between relational databases (RDBMS) and Neo4j[13].

shows that graph databases handle data relationships with high efficiency.

TigerGraph (DBMS Property Graph Model f) outperformed Neo4j by more than 100 times faster in some instances, and that become bigger with the scaling of data more.[15]

3 The definition of KG:

3.1 Tracking the term KG in the history

According to [16, 17], the concept was broadly focused on a year 2012 when google introduced this concept to the world by marketing her KG and get followed by other companies IBM [18], LinkedIn [19], eBay [20]; which led to the emergence of a wide range of definitions and works on, as we see 7 definitions before 2012 and 24 definitions after as we see Table 1and Table 2, and a large number of diverse research on this topic like fig. 2 show. But the term itself dates back to the 1970s according to[17] and for more precision 1972 [21] in computer science context, but first use for problem solving in modern history was in 1736 by Leonhard Euler to solve the famous Seven Bridges of Königsberg problem[22], although the idea of presenting the information and concept in graph model is older than that, [23]says: "The idea of representing, or at least displaying, knowledge as a graphical diagram (rather

than as, say, a set of sentences) has a very old history. In its modern sense it goes back at least to 1885 (CS Peirce ‘existential graph’) and can probably be traced into medieval writings and earlier. (The Torah version of Genesis refers to a ‘tree of knowledge’.) It has been re-invented or rediscovered many times since, and seems to blossom in public (or at least academic) discussions with a periodicity of roughly 40 years.".

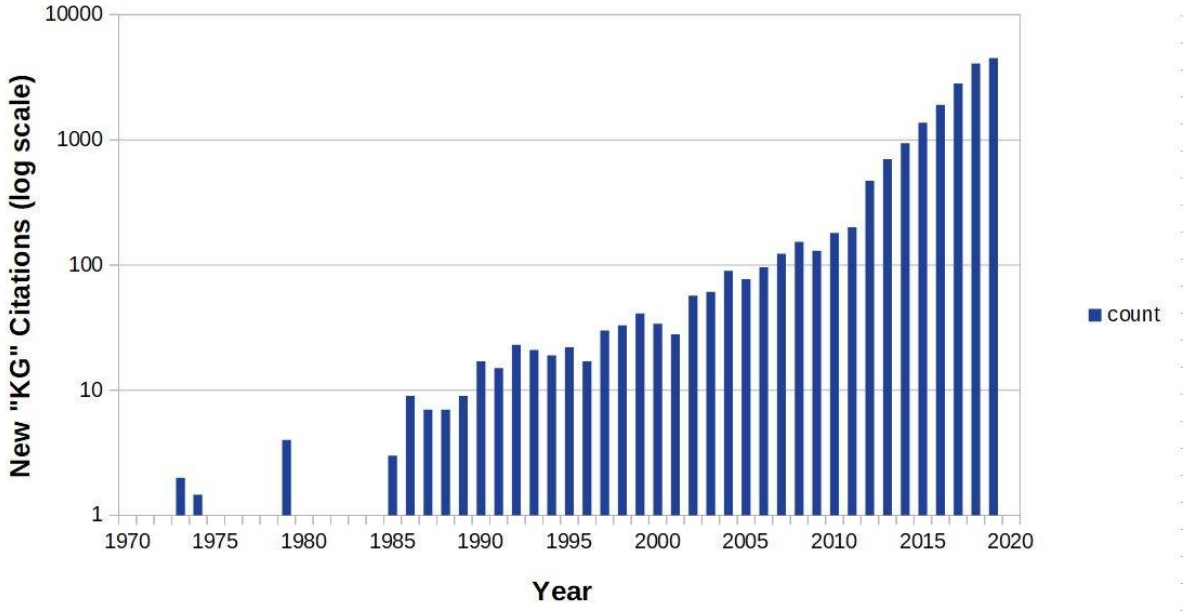

fig. 5.KG Citations, 1970 – 2019[17]

3.2 Accumulation definitions for knowledge representation in graph

Before beginning the explanation, start with some definitions by scientists in the field collected by[17] based on other work like Ehrlinger and Wöß [16], Gutierrez, and Hogan et al..., the two tables look like:

Table 1.Selected Definitions of KG over time[17].

Source	**Year**	**Definition**
Marchi and Miquel	1974	A mathematical structure with vertices as knowledge units connected by edges that represent the prerequisite relation
Hoede	1982	See [24] and text and Bakker and de Vries below
Bakker	1987	A knowledge graph is a labeled directed graph D(P,A) with (a) P is a set of points that represent concepts, relations or frameworks; (b) $A \subseteq P \chi P$ is a set of arcs that form the connections between the entities. An arc exists

		for $p_1 \in P$ to $p_2 \in P$ if: (b1) p_1 represents a concept that is the tail of a relation represented by p_2; (b2) p_2 represents a concept that is the head of a relation represented by p_1. (b3) p_1 represents an element in the contents of a framework and p_2 represents the FPAR-relation. (b4) p_2 is a framework and p_1 represents the accompanying FPAR-relation
de Vries	1989	A directed graph that distinguishes three types of asserted relationships: (1) an object has a certain property, (2) an object is an instance of another object, (3) a change in a property of an object leads to a change in another property of that object
James	1992	A knowledge graph is a kind of semantic network. . . . One of the essential differences between knowledge graphs and semantic networks is the explicit choice of only a few types of relations
Zhang	2002	A new method of knowledge representation, [which] belongs to the category of semantic networks. In principle, the composition of a knowledge graph is including concept (tokens and types) and relationship (binary and multivariate relation)
Popping	2003	A particular kind of semantic network
Singhal (Google)	2012	A graph that understands real-world entities and their relationships to one another: things, not strings
Ehrlinger and Wöß	2016	A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge
Krötzsch and Weikum	2016	Knowledge graphs are large networks of entities, their semantic types, properties, and relationships between entities
Krötzsch	2017	Are characterized by several properties that together distinguish them from more traditional knowledge management paradigms: (1) Normalization: Information is decomposed into small units of information, interpreted as edges of some form of graph. (2) Connectivity: Knowledge is represented by the relationships between

		these units. (3) Context: Data is enriched with contextual information to record aspects such as temporal validity, provenance, trustworthiness, or other side conditions and details
Paulheim	2017	A knowledge graph (i) mainly describes real world entities and their interrelations, organized in a graph, (ii) defines possible classes and relations of entities in a schema, (iii) allows for potentially interrelating arbitrary entities with each other and (iv) covers various topical domains
Shao *et al.*	2017	A knowledge graph is a graph constructed by representing each item, entity and user as nodes, and linking those nodes that interact with each other via edges.
Villazon-Terrazas *et al.*	2017	A knowledge graph is a set of typed entities (with attributes) which relate to one another by typed relationships. The types of entities and relationships are defined in schemas that are called ontologies. Such defined types are called vocabulary. A knowledge graph is a structured dataset that is compatible with the RDF data model and has an (OWL) ontology as its schema.
Wilcke et al.	2017	A data model used in the Semantic Web . . . based on three basic principles: 1. Encode knowledge using statements. 2. Express background knowledge in ontologies. 3. Reuse knowledge between datasets.
Bianchi et al.	2018	- Large knowledge bases - Entities classified using types - Types organized in sub-types graphs - Binary relationships between entities - Semantics and inference via rules/axioms - Semantic similarity with lexical, topological and other feature-based approaches.
McCusker *et al.*	2018	An Unambiguous Graph [a graph where the relations and entities are unambiguously identified] with a limited set

		of relations used to label the edges that encodes the provenance, especially justification and attribution, of the assertions.
Xiong	2018	A data resource that contains entries ('entities') that have their own meanings and also information ('knowledge graph semantics') about those entries.
Bellomarini *et al.*	2019	A semi-structured datamodel characterized by three components: (i) a ground extensional component, that is, a set of relational constructs for schema and data (which can be effectively modeled as graphs or generalizations thereof); (ii) an intensional component, that is, a set of inference rules over the constructs of the ground extensional component; (iii) a derived extensional component that can be produced as the result of the application of the inference rules over the ground extensional component (with the so-called "reasoning" process).
d'Amato *et al.*	2019	A graph-based structured data organization, endowed with formal semantics (*i.e.*, a graph-based data organization where a schema and multiple labelled relations with formal meaning are available).
Columbia University	2019	An organized and curated set of facts that provide support for models to understand the world.
Hogan *et al.*	2019	A graph of data with the intent to compose knowledge.
Kejriwal	2019	A graph-theoretic representation of human knowledge such that it can be ingested with semantics by a machine; a set of triples, with each triple intuitively representing an 'assertion'.
Morrison	2019	A knowledge base that's made machine readable with the help of logically consistent, linked graphs that together constitute an interrelated group of facts.
PoolParty	2019	A graph-theoretic representation of human knowledge such that it can be ingested with semantics by a machine; a set of triples, with each triple intuitively representing an 'assertion'.
Tran and Takasu	2019	A collection of triples, with each triple (h,t,r) denoting the

		fact that relation *r* exists between head entity *h* and tail entity *t.*
Vidal *et al.*	2019	A knowledge graph is presented as the intersection of the formal models able to represent facts of various types and levels of abstraction using a graph-based formalism.

Table 2 Wöß definitions that were not mentioned by the Bergman, in addition to the definition of Bergman, we added it.

Source	**Year**	**Definition**
Pujara et al. **[25]**	2013	[...] systems exist, [...], which use a variety of techniques to extract new knowledge, in the form of facts, from the web. These facts are interrelated, and hence, recently this extracted knowledge has been referred to as a knowledge graph."
Faarber et al. [26]	2016	We define a Knowledge Graph as an RDF graph. An RDF graph consists of a set of RDF triples where each RDF triple (s; p; o) is an ordered set of the following RDF terms: a subject $s \in U \cup B$,a predicate $p \in U$, and an object $U \cup B \cup L$. An RDF term is either a URI $u \in U$, a blank node $b \in B$, or a literal $l \in L$. U, B, and L are pairwise disjoint. We denote the system that hosts a KG g with h_g.
Andreas Blumauer **[27]**	2016	Knowledge graphs could be envisaged as a network of all kind things which are relevant to a specific domain or to an organization. They are not limited to abstract concepts and relations but can also contain instances of things like documents and datasets
Michael K. Bergman[17]	2019	Thus, knowledge graphs, like ontologies, have a broad range of applications and constructions. When one hears the term, it is less important to reflect on some precise understanding as to realize that human language and knowledge is being presented in a connected, graph form.

		That, I believe, is the better practical and common-sense understanding of KGs.[1]

As we saw there are differences, even contradictions, for example some consider it simply as nodes represent entities, and edges represent relationships between those entities, but they were criticized as not being different from graph (database); author define it as graph-structured knowledge base, so we go back to a point further represented in the ambiguities have also existed regarding the definition of a “knowledge base” (KB); others tried to detail in the definition and set specific standards like Paulheim, so they failed where excluded famous KGs like Geonames -because it does not accept domain specific graphs-.

In light of all this, the promoters of this technology such google, Freebase, DBpedia provide only a general description without going into details.

After broad embrace by the semantic Web community earnest about 2016 been a concerted effort to provide more “precise” understandings of what the KG concept(A few conferences have been devoted).[17]

[17] explains this dispute as follows:" Attempts to provide a “precise” definition of KGs appear likely doomed to failure, and appear to be more a reflection of prior prejudices than a derivation of some discovered inherent meaning."

3.3 Bedrock of KG definition

Finally, at the end the concept is one, but different angle of view in how its description.

We can express KG with what is generally shared between existing applications and written definitions, so we can define like this:

The KG is a powerful data structure that express the data we interest with a directed graph $\mathcal{G}$ where $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$ with $\mathcal{E}$ it is a set of entities, $\mathcal{R}$ set of relations and $\mathcal{T}$ facts (assertion) that represented in the form (h, r, t) so that it indicates there is a relation r ∈ R between h (header or subject) and t (tail or object) where h, t ∈ $\mathcal{E}$. Until this point of definition we talk about data graph and not KG when we attached it with a schema, identifiers and context we can talk about KG- we will explain each of them in the remainder of the theses-, this attachment allow us to look for inference and extract new knowledge.

[1] He explained his concept further in ations[17] Bergman, M. K. A COMMON SENSE VIEW OF KNOWLEDGE GRAPHS (July 1, 2019)..

The KG is powerful because the reasons below (Split the items for [28] but explain it from multiple sources):

1 - It's a graph (data structure view): the KG have the graph property when make it easy to apply graph algorithm (short path, coloring ...).

2 - It's semantic: self-descriptive the meaning in the graph itself the naming of nodes and edge should be like the real world, so from there the queries are close to natural language (use natural language vocabulary like we see in the official ontologies.

the final result of what we mentioned it, is:

- whiteboard-friendly[1]: no need for modifying the data model to fit a normalized the DB, the graph data model stays exactly as it was drawn on the whiteboard -In other word no need for model transformation-.

- Make the discovery and search smarter and more efficient.

3 - It's smart: can be solid structure for machine learning (inductive reasoning), and the logical why can we describe what make the why for deductive reasoning.

4 - It's alive: That because KG can store the metadata for the information such as provenance, versioning information. what make it good too deal with dynamic dataset. And that make it useful tool to define the credibility for final users.

In short sentence "A knowledge graph can answer what it knows, and also how and why it knows it"[28]

resource used [16, 28, 29] [30].

Note about the term KB: There is some kind of ado about use it interchangeably with KG, which some consider it a problem "... misleading assumption that the term knowledge graph is a synonym for knowledge base, which is itself often used as synonym for ontology."[16], we did not find a source who puts a finger on the wound ; but in practice the research projects and papers go ahead we tend to term KG because it expresses structure and make it explicit especially this thesis.

[1] when designing a data model, people draw example data on the whiteboard and connect it to other data drawn to show how different items connec.

3.4 Graph data models and the use cases

In this section, we introduce three most popular types of graph data model, with an explanation of the use case of each one, the other type surveyed by [1], after move to the use case for each, the general use case for the graph is when the importance of relation and interconnectivity as the same level of the data or more[1]:

3.4.1 Edge-labelled graphs

Also referred to it as triple stores, RDF store or semantic graph database[1].

Edge-labelled graphs present Tim Berners-Lee, Jim Hendler and Ora Lassila overview of how web should be [31].

Edge-labelled graphs (RDF graphs) are static snapshots of information. Some RDF graphs sources may be immutable snapshots of another RDF source, archiving its state at some point in time, relations that involve more than two entities can only be indirectly expressed in RDF [32]. It mainly used for semantic web duo his data exchange orientation, the correction framework that we work on base on that model of graph.

The core structure of the edge-labelled graphs is a set of triples (assertion) that represented using RDF framework where each consisting of a subject, a predicate and an object, and querying via SPARQL.

RDF for Edge-labelled graphs:

This model use RDF (Resource Description Framework) as standard recommended by the W3C to encode the web content to be machine-readable, it represent data as triples, each entities -nodes- (subject, object) represented by URIs, blank nodes[2], or datatype literals and the predicate -edges- it is a URI to relate them; that what provide the linking; to come up with a facts database in graph form[33, 34].

Any IRI or literal: reflects a thing -resources- in the world (the "universe of discourse") physical things, abstract concepts.

- The resource denoted by an IRI (Internationalized Resource Identifiers) it is called its referent.

---

[1] Here we refer to original source "they call themselves “semantic graph database.” "[31] Barrasa, J. RDF Triple Stores vs. Labeled Property Graphs: What’s the Difference? (18 aug 2017).

[2] Blank node (also known as bnode, anonymous resource) it is a node without URI or literal.

- The resource denoted by a literal is called its literal value. Literals have datatypes that define the range of possible values (strings - can attached with language-tagged-, numbers, and dates).

- Blank nodes do not identify specific resources, the statements involving blank nodes say that something with the given relationships exists, without explicitly naming it [35].

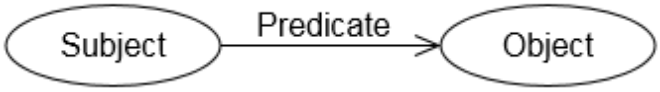

fig. 6 RDF graph can be visualization[1].

The RDF store give some advantages:

1 - Simple to Import/Export: due to standardization formats N-Triples or N-Quads[2] by RDF. then easy consistency when merging because of we can URI can generate uniform vocabulary[3].[33]

2 - Inference potential (reasoning): when get attached by ontologies and the atomic way the triples stores represent the data, give it more potential for determine the semantic of things with more specific meaning, that is good for machine Learning algorithms to minimize disambiguate, and for Semantic Analysis (text mining), without forget his strong support to detective knowledge extraction, and the no need for exact matches what open the door for intelligent recommendation engines, search term expansion knowledge … etc.(in this way solid especially if the datasets slow-changing - if not immutable - will give more than other KG models) [33] [31] [36].

- RDF stores are very strongly index-based.

This way of storing in this model allowed for the cited features, but also be hindrance for other application such: transactional applications that require updating sets of data.[33] and also The lack of property makes it difficult to represent complex relations fig. 9, which is reflected in the query later as well, we mention that the transition to/from directed edge-labelled graphs and/or graph datasets can done without lose any information[12].

An example shows the impact of this deficiency in relationships, ex form Neo4j.com:

[1] An RDF graph can be visualized as a node and directed-arc diagram, in which each triple is represented as a node-arc-node link.
[2] Set of statements where it is part of in a dataset (a graph) identified by a URI and can have label (called also named graph).
[3] This is a big advantage of an analysis software that needs to combine multiple sources.

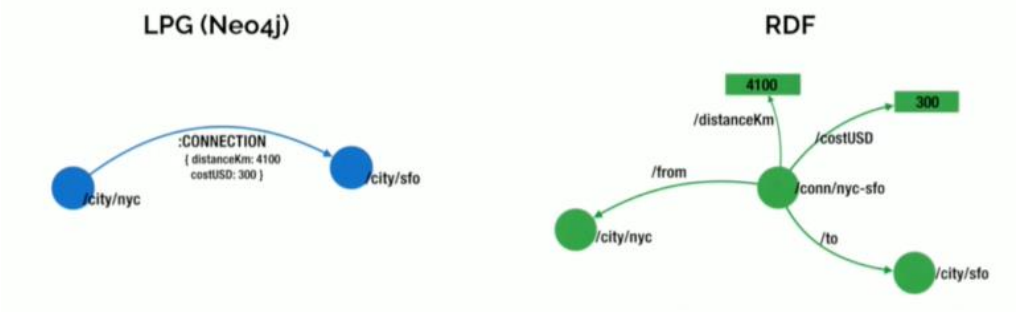

fig. 7 Comparison of the effect of relationships of correlation in simple modeling.

We mention the can be merged with other DB (the source), create smart Metadata that describes the contents of the relational database.[33]

Note: An RDF graph can be visualized as a node and directed-arc diagram.

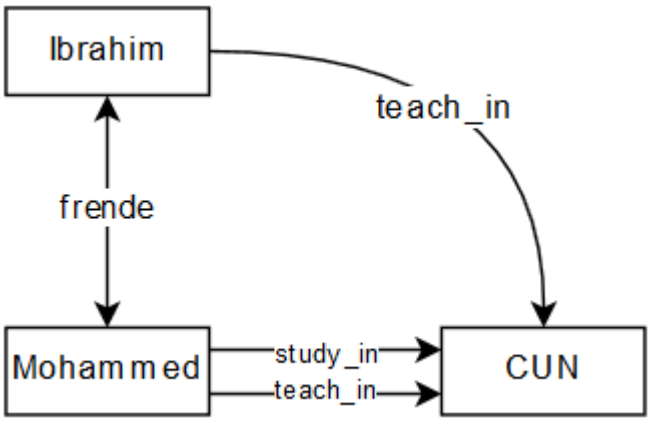

fig. 8 graph can visualization example.

The Query Language for RDF (SPARQL):

The standard W3C query language for RDF it is SPARQL, we can query with it in many manners [33]:

1 - SELECT: Returns all, or a subset of, the variables bound in a query pattern match.

2 - CONSTRUCT: Returns an RDF graph constructed by substituting variables in a set of triple templates.

3 - ASK: Returns a boolean indicating whether a query pattern matches or not.

4 - DESCRIBE: Returns an RDF graph that describes the resources found.

RDF dataset:

An RDF dataset is a collection of RDF graphs. all but one of these graphs have an associated IRI or blank node. They are called named graphs, and the IRI or blank node is called the graph name. The remaining graph does not have an associated IRI, and is called the default graph of the RDF dataset.

There are many possible uses for RDF datasets. One such use is to hold snapshots of multiple RDF sources. like a side effect the marge of many resources can update or refine data and determine the level of trust for each source what is required in the semantic web.

3.4.2 label Property Graph Model:

When Tim Berners and his team were described their vision of the internet, around this time a group of Swedish engineers were developing an enterprise content management (ECM) system by using data graph model to ending by the labeled property graph and later became Neo4j.[31]

We tell property graphs, where nodes and edges can further have attributes, each edge and node are additionally associated with a unique identifier which provide extra metadata from the pair's property–value-.

The Property Graph Model can be also:

- Multi-valued property graphs: allow multiple labels and multi-valued attribute properties.

- Custom: depending on the database management system developer; for example, Neo4j allows only one label on each edge, multiple labels on nodes, and one value on each attribute property (albeit potentially a list).

Some characteristics:

- Index-free adjacency: the connections stored in the physical disc; so while searching it search for pointers and not scanning indexes [37].[31] what provide rapid traversal of data connections.[36]

- Provide additional flexibility when modelling more complex relations.

So label property graph is mainly focus/oriented on storage and querying.[31] and give huge value for operational and transactional use cases.

3.4.3 The Hypergraph Model (HDM):

The Hypergraph model will allow multiple nodes to be connected by the same relationship - relationship called a hyperedge in the Hypergraph -.

Use for meta-intent For example [36], if you need to qualify one relationship with another (e.g., I like the fact that you liked that car. Whoa, so meta.).

Can always represent a hypergraph as a property graph with adding more relationships and nodes, but not the revers[36].

Example for Hypergraph is Grakn.ai.

The explicit data model make some confusion and hard time while in developing.[36]

The Hypergraph and property graph they isomorphic[1].

The number of relationships decreases dramatically

3.5 Graph feeding

Multiple sources ways to feed the graph as cited below depending on classified depending on accuracy convert:

1 - Text Sources: Text corpora (newspapers, books, scientific articles, social media, emails ...etc.) it's rich information sources, but the process to extract it and enrich the graph the KG with is not trivial task, to this end, we need technologies of Natural Language Processing

(NLP) and Information Extraction (IE) in order to extract the right entity and right relation and like it to the KG [38].

2 - Markup Sources: HTML, Wikitext used by Wikipedia, TeX for typesetting ... etc.; by strip the markers (e.g., HTML tags), leaving only plain text and text Sources reuse steps this time the markup can make the steps easier [38].

3 - Structured Sources/tree-structured: Structured Sources such relational databases [39], CSV files ... etc., and tree-structured by hybrid query languages such as XSPARQL [40], both can be converted into KG precisely.

[1] can convert a Hypergraph to a property graph and vice versa (Hypergraphs to property graph adding more nodes, and the reverse for the property graph).

4 Form data graph to knowledge graph

Like mentioned before data graph contains just the data like presented in the definition, to go with this data to KG we need to enhance that graph with schema, identifiers and context, here discover these concepts, the section base on [27] [41] [12] [32] [42] [34] [35] [43] [44]

4.1 Schema

Like we discussed one of the KG powers how it's schema flexible is, to give us the choice to define it in the time we need to of the project life or even not.

Schema used for define the high-level structure and/or semantics that the graph follows or should follow.

Three type of schema can be defined: semantic, validating, and emergent.

4.1.1 Semantic schema

It's provides a data-modelling vocabulary to define the meaning of high-level terms (vocabulary, terminology); what provide a base to facilitates the reasoning over the graph.

The definitions can be done for both classes and properties. Many techniques through which the meaning of classes and properties can be described to capture meaningful generalizations about data in the web: RDF Schema, ontology languages (too richer vocabulary) such as OWL.

The usual technology to define the semantic schema for a triple store graph it by using is RDF vocabulary extension called RDFS [45] with the using of subclasses, subproperties, domains, and ranges amongst the classes and properties, if we need to enforce this embedding of the graph with OWL will provide much more semantic values [46] we talk about it later in this section.

Semantic schema more convenient for OWA in the reason to get the information that are not declared explicitly and typically defined for incomplete graph data (what we don't know can be exit the contrary of CWA).

We are talking briefly about OWL, which we need later:

OWL have high capability to define the terms precisely depending:

The quality of the ontology depending on level of details required (not less no more to manage the complexity) and consistently, those factors guide the automate entailment in the right way if done good.

The OWL language provides three increasingly expressive sublanguages designed for different use case definition for [46]:

OWL Lite: supports those users primarily needing a classification hierarchy and simple constraint features.

OWL DL: supports those users who want the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems.

OWL Full: is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.

Classes:

The resources it's groups of classes and the classes themselves resources, identified by IRIs or described by using RDF properties. the instances of the class are a member of a class. to refer that a resource is an instance of a class we use the *rdf:type* property.

A class may be a member of its own class extension and may be an instance of itself.

The technical definition blow of classes from tdwg platform[1] and from [46] for owl, we take what we need:

rdfs:Resource:

All things described by RDF are called resources, and are instances of the class *rdfs:Resource*. This is the class of everything. All other classes are subclasses of this class. *rdfs:Resource* is an instance of *rdfs:Class*.

rdfs:Class:

This is the class of resources that are RDF classes. *rdfs:Class* is an instance of *rdfs:Class*.

rdfs:Literal:

The class *rdfs:Literal* is the class of literal values such as strings and integers.

[1] https://terms.tdwg.org

rdfs:Literal is an instance of rdfs:Class. *rdfs:Literal* is a subclass of rdfs:Resource.

rdfs:Datatype:

property:

The property e used to state that a resource is an instance of a class, the RDF property it a relation between subject resources and object resources.

This specification defines the concept of subproperty. The *rdfs:subPropertyOf* property may be used to state that one property is a subproperty of another. If a property P is a subproperty of property P', then all pairs of resources which are related by P are also related by P'. The term super-property is often used as the inverse of subproperty.

rdfs:subClassOf:

The subject is a subclass of a class.

Constraints in the context of RDF Schema:

rdfs:domain: rdfs:Class (=the class, to which rdfs:subClassOf may be applied).

rdfs:range: rdfs:Class (=the class or datatype of the values or objects that are assigned to "rdfs:subClassOf", i.e. the right side in the triple).

owl:DatatypeProperty:

Describe type of values a triple with the property should have, where it relates individuals with literal.

owl:ObjectProperty:

Describe type of values a triple with the property should have, where it relates individuals with other individuals.

FunctionalProperty :

If a property, P, is tagged as functional then for all x, y, and z:

P(x,y) and P(x,z) implies y = z

rdfs:label:

label: Is an instance of rdf:Property that may be used to provide a human-readable version of a resource's name.

The meaning of ontology:

The interpretation it is mapping the nodes and edges in the data graph to nodes (nodes/edges) and edges of the domain graph (entities/relations) where the last it is the meaning of ontology features and entailment (ex: default schema).

The interpretation its guild by the general assumptions (UNA, NUNA, OWA, CWA).

Semantic define:

There two elements for interpretation:

1 - Domain graph.

2 - A mapping from the terms (nodes and edge-labels) of the data graph to those of the domain graph.

Where the domain graph has the same model as data graph (ex the both are directed edge-labelled graph).

We tell the interpretations it is satisfy the data graph if it respects the assumptions and semantic conditions.

OWL support condition and feature for individuals (same as, diff. from ...etc.), and properties (define: equivalent, disjoint...etc.), Classes (hierarchies using sub-classes, disjoint... etc.), provide metadata about ontologies (versioning ..), feature to violate the assumption legally[1]; for more details we refer to the OWL 2 standard [44].

4.1.2 Default semantics (Assumptions)

Closed World Assumption (CWA):

Assumed all positive information of the world has been specified in other word the data graph is described the world completely, then conclude any positive fact that has not been specified or cannot be inferred from this information is it false.

Open World Assumption (OWA):

Is the opposite concept for CWA where, assume our knowledge of the world it incomplete so therefore any information not explicitly specified or cannot infer from the KG it's considered unknown and not false.

[1] Define a key for a class, denoting the set of properties whose values uniquely identify the entities of that class. Without adopting UNA.

Local Closed World Assumption (LCWA):

In many situation we be It is unacceptable to assume that the system has full knowledge and at the same time is not able to unable to definitely answer “yes” or “no” (when the information it missing).so the solution it mirage the both CWA and OWA where we assumed a portions of the data graph are complete.

Unique Name Assumption (UNA):

Different names always refer to different entities in the world.

No Unique Name Assumption (NUNA):

Reverse of Unique Name Assumption.

4.1.3 schema Validating

In many cases weed some minimum of explicit information with clear constraint in the graph -we talk about -, here we use the validation schema to make this part of the graph complete.

To define the validating schema, we usually use the standard "shapes ". the idea is to target the needed nodes to be exist in specific manner - e.g.: Instances of a class ... etc.- before the extraction we then define the necessary constraints on the nodes and define their values; to get finally graph called shapes graph from interrelated shape nodes; represent his graph in UML-like class diagrams.

Even when we need to make some part of it close can be flexible, because the shapes can be open, the open shapes let modeler add properties for those nodes that he don't know in the modeling phase; in the other hand we have the notion of close when that prohibit that.

shapes languages provide some tools that give more capability for the modeler like Boolean features (AND, OR, NOT) and recursion, but in the other side can cause semantic problems like logical paradox -there may these, there some paper on that-.

Two shape languages have recently emerged for RDF graphs: Shape Expressions (ShEx), published as a W3C Community Group Report [47]; and SHACL (Shapes Constraint Language), published as a W3C Recommendation [48].

4.1.4 Interaction between schema and validating schema

validating schema and semantic schemata play different role, where semantic schema allow for inferring new data, validating schemata allow for validating existing graph data (

including inferenced data), they can complement each other by define constraints in a validating schema and validate the data graph w.r.t the schema results, and listing constraint violations.

4.1.5 Emergent schema

Both semantic and validating schemata need the expert of the domain to define, but the emergent schema can be extracted automatically, because any data graph good builded it have some hidden structure inside.

The goal to extract the emergent schema is to give an overview of the data graph for humans, in order to define semantic or validating schema, optimizing the indexing and querying.

There many frameworks to extract emergent schema for one of those bisimilarity quotient graphs.

4.2 Identifying things in KG

We use the identifier on entities to disambiguate the terms to avoid naming clashes w.r.t. an external source spicily when we look to extend our graph.

Here we talk how to identify things on KG.

4.2.1 External identity links:

The previous technique we see not enough to make the terms have the same meaning for all graphs (when look at merge graphs); some we soak in define the same terms with different URI of these graphs used different namespacing.

There two solution for this issue:

1 - Enforce the entity with additional piece of information: that can help to distinct entities when merge the graphs (e.g. postal code, the year it was founded ...etc. it helps when matching).

2 - Identity links: it means those entities has the same semantic identity in other external resource. this concept supported in OWL standard *as owl:sameAs* property.

4.2.2 Datatypes:

The syntactic of data types known (decimal, Boolean, integer, date Time ...etc.), it refers to it meaning so no need to give it IRIs and also it is recognizable by the machine (order values, extract the dates ... etc.).

To identify those data types RDF support XML Schema Datatypes (XSD)[43], where a datatype node is given as a pair (l,d) where l is a lexical string, and d is an IRI denoting the datatype, and the default data type of the d not define will assumed as string.

we called these nodes literals, note that are not allowed to have outgoing edges.

4.2.3 Lexicalisation:

The Lexical of the terms can be interpretable for humans [Animal:lion] , but in in the most of time in semantic web the global identifiers used to identify the things to the machine not us, if take example the potato defined as wd:Q16587531 in Wikidata, that make the knowledge independent on the language and the instance the time (in case the name of the entity it changed in the real world).But that doesn't mean we ignore the humans, at last the application used to contact the humans at last of the process for that, many time in the real name in the real world; for this purpose RDFs provide option to make it human-interpretable by adding labels for there comments, aliases and anchor to refer to the entity linguistically and the label can be enhanced with land tags with give more value in some graph to present that knowledge more efficient for the language targeted, example below from DBpedia.

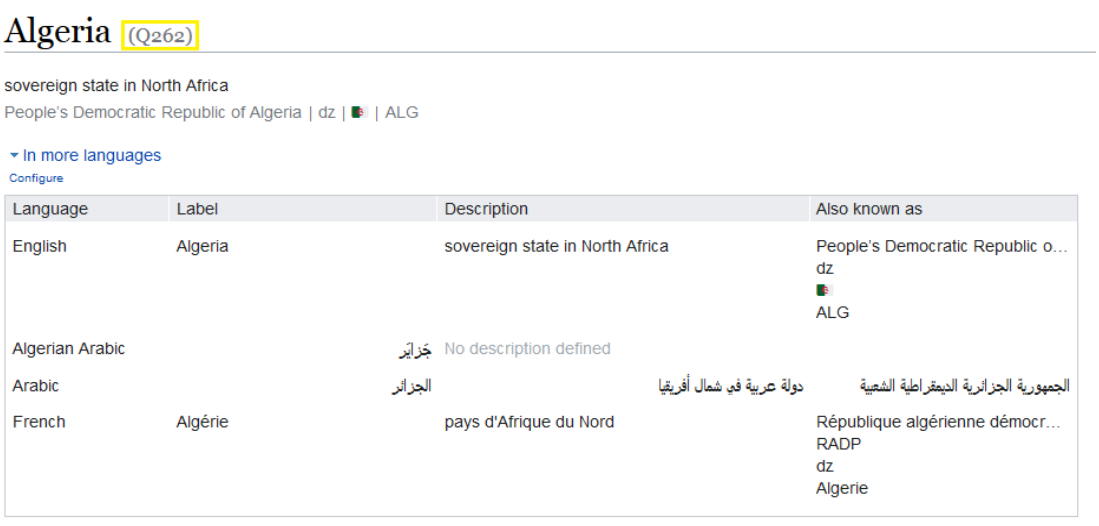

fig. 9. Lexicalisation.

4.2.4 Existential nodes:

In case we know the existence of the node but we can't identify with the given relationships exists we called it existential mood, existential nodes are supported in RDF as blank nodes, but complicate operations on graphs.

4.3 Context

When we define the context, we define what called truth scope of we give as the reference to tell if certain fact is true or not. for that we make the context explicit to give us the possibility to define a more efficient truth scope.

5 (semi-)automatically in the KG

Because we aim to do (semi-)automatically process we need to talk about that.

KG have to intelligent method to deal with data deductive and inductive.

we talked briefly about deductive and more details form inductive because the framework using the ML.

5.1 Deductive

The knowledge can be the commonsense knowledge its general premises and rules that shard by the public, and domain knowledge it known among expert only [mention to medal db of the application], this why to learn (premises + rules -> result not know before) it one of the methods that human use to gain implicit knowledge and it the deductive method.

The machine needs formal instructions to define the premises and rules to simulate the human deductive knowledge extraction to reach the conclusion that surpasses the human processing capability in terms of size, precision and efficiency.

This type of thinking can be applied in: (deductive) classification, finding inconsistencies ... etc.

The complex entailments can be expressed and automated formally defined via ontologies.

[12]

Some consistency works based on that reasoning type.

5.2 Inductive

Deductive knowledge flow a logical consequences, inductive knowledge base on the observations of the inputs to generate predictions that potentially can be imprecise (the value of confidence can be associated more often) but that does not diminish its value when used in the right place (usually for encode patterns or predictions) that may outperform the experts.

We can inductively from graphs with supervised or unsupervised method.

Supervised methods learn a model to map a given set of example inputs to their labelled outputs; after the model learn it able to predict output for unlabeled inputs; but manual labelling can be costly so here we can look to the self-supervision by generate the input–output pairs automatically from the input literals .

Unsupervised processes do not require labelled input–output pairs, but apply a predefined function (typically statistical in nature) to inductive the output [map inputs to outputs].

There to many works on many ways for knowledge graph to exploit the inductive reasoning here we cover just knowledge graph embeddings What has related to the next chapter, then the big family on that. the section inspired from [49] [50] [51] [52] [53] [54] [12] [55].

5.2.1 Knowledge Graph Embeddings:

Machine learning is the face of the inductive knowledge, and the graph can exploitation it power to refining the knowledge graph itself (predicting new edges and/or identifying erroneous edges) or downstream tasks (train models for classification, recommendation, regression).

Usually the input of the traditional machine learning it is a numerical vector but the graph doesn't have that numerical nature to transform the graph to vector easily, but there work on that problem, what we will look for that.

The early trying of convert a Edge-labelled graphs to vector was by using one-hot encoding, where generating a vector for each node of length $|L| \cdot |V|$ -where $|V|$ the number of nodes in the input graph and $|L|$ the number of edge labels- placing a one at the corresponding index to indicate the existence of the respective edge in the graph or zero if else; but that put us in the situation where you get large and sparse vectors what is not good for machine learning models.

The knowledge graph embedding techniques strive to represent the graph in a dense and continuous, low-dimensional vector space to used by it for machine learning.

The dimensionality d of the embedding is fixed usually low.

The graph embedding generally composed of an:

- Entity embedding (we denote by e) for each node: a vector with d dimensions.

- Relation embedding (we denote by r) for each edge label: (typically) a vector with d dimensions.

This vectors used for abstract with preserve the characteristics of the graph structures this notion of embedding can be exploited in many ways, usually by using a scoring function (e.g. TransE) that accept e_s (embedding of subject), e_o (embedding of object) and r_p (embedding of edge label p), to figure out the plausibility of the assertion: how likely to be true, for final goal to maximize the plausibility of positive assertion (typically edges in the graph) and minimize the plausibility of negative examples (typically edges in the graph with a node or edge label changed such that they are no longer in the graph) the result depending on the how function defined; models learnt through self-supervision to output plausibility scores by encoding features of the graph.

The embedding graph it is useful in many tasks: can use the plausibility scoring function to assign a confidence for assertion (e.g. assertion from external source), complete edges with missing nodes/edge labels for the purposes of link prediction and finding duplicate nodes that refer to the same entity[1], which can be used in turn for recommendations system.

There too many work on this subject "graph embedding techniques", many get Surveyed by [54].

There some of family for embedding: translational models, tensor decomposition models, neural models.

Translational models

Translational models interpret edge labels as transformations from subject nodes to object nodes for explicit or implicit relation, for example in the graph below the relation Bus transforming A to C and also B to D.

[1] By assign similar vectors to similar nodes and similar edge-labels.

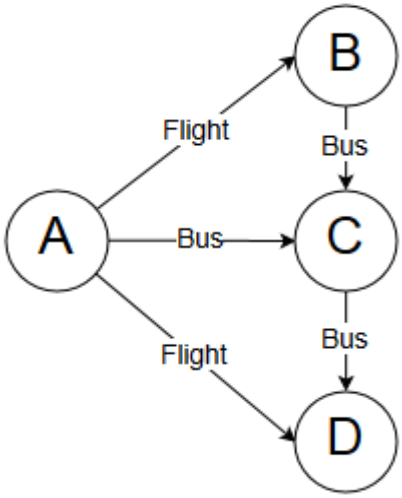

fig. 10 simple graph.

The base approach for this family is TransE [53]. The why it works for all positive edges <s,p,o>, TransE learns vectors e_s, r_p, and eo aiming to make $e_s + r_p$ close possible to e_o. conversely, if the edge is a negative example attempts to keeps e_o away from e$e_s + r_p$.

The translational be used for to predict edges (among other tasks).

TransE will, aim to generalize and give similar vectors to all target nodes, which may not be true in all case, assign cyclical relations a zero vector as the directional components will tend to cancel each other. to resolve this issues, many work process such TransH [51] that use represents different relations using distinct hyperplanes, where for the edge <s,p,o> , s is first projected onto the hyperplane of p before the translation to o is learnt ,TransR [51] generalizes this approach by projecting s and o into a vector space specific to p, which involves multiplying the entity embeddings for s and o by a projection matrix specific to p. generalizes, TransD [51] simplifies TransR by associating entities and relations with a second vector, where these secondary vectors are used to project the entity into a relation-specific vector space; other translational models in the survey [54].

5.2.2 Tensor decomposition models

Tensor decomposition look to decomposing the tensor into more “elemental” tensors where the original tensor can be recomposed (approximated) with a fixed sequence of basic operations; the element tensors decomposed in the way tried to capture latent factors on the original tensor for each. one of the models based on that approach is DistMult [50] where for each entity and relation is associated with a vector of dimension d, such that for an edge <s,p,o> we have plausibility scoring function $\sum_{i=1}^{d}(e_s)_i\,(r_p)_i(e_o)_i$ where $(e_s)_i$, $(r_p)_i$ and

$(e_o)_i$ denote the i^{th} elements of vectors e_s , r_p , e_o , respectively, so goal is learn vectors for each node and edge label that maximize the plausibility of positive edges and minimize the plausibility of negative edges; the down side of this approach it does not consider edge direction. there some method to solve that issue but with height complexity such RESCAL.

5.2.3 Neural models

Both of method above are linear or bilinear what limit the models for solve linear problems, so community look for adapt that with non-linear problems one of the proposed solutions it using neural networks to learn embeddings with non-linear scoring functions for plausibility.

We use Multi Layer Perceptron (MLP) [56].

5.2.4 Language models

the first use is to represent the words for machine learning such word2vec [52] where it compute embeddings for words based on large amount of text with neural networks trained, where that words used in similar contexts (e.g., “flag”, “banner”) have similar vectors, and used to file the gaps (predicting); from that come RDF2Vec [49] that use random walks on the graph and records the paths (the sequence of nodes and edge labels traversed) as “sentences” then put it in word2vec like input.

6 Exist KG

6.1 Open source (open knowledge graphs):

6.1.1 Dbpedia

Dbpedia it is the structured form of Wikipedia with automatic extraction and provides public access to users and companies for business intelligence & analytics, entity extraction, natural language processing, reasoning & inference, machine learning services, and artificial intelligence in general.

Now we present some statistic about dbpedia from official site[1]: "Altogether the DBpedia 2014 release consists of 3 billion pieces of information (RDF triples) out of which 580 million were extracted from the English edition of Wikipedia, 2.46 billion were extracted from other language editions." and support 125 languages.

[1] https://wiki.dbpedia.org

6.1.2 Yet Another Great Ontology (YAGO):

YAGO[1] it a KG derived from Wikipedia, WordNet, WikiData, GeoNames, and other data sources. Contain more than 17 million entities (like persons, organizations, cities, etc.) and contains more than 150 million facts about these entities, the statistic taken from their official github page[2].

6.1.3 Other:

There other cross-domain knowledge graphs Wikidata, Freebase, BabelNet ...etc.; and other are specific domains, like BBC World Service Archive, OpenCitations...etc.

6.2 Closed source (enterprise knowledge graphs)

6.2.1 Product Graph (Amazon)

"The Product Graph team uses a variety of machine learning techniques to get product-related information from Amazon detail pages and the Internet at large."[57]

6.2.2 EbayShopBot

understand the consumer (Which use general expressions, ex"I want a bag") by interactions to provide context and find richer meaning in conversational-style searches to give him a result that he desires over 1.1 billion live listings (2017 stat).

ShopBot uses its Knowledge Graph (Neo4j) to understand user requests and generate follow-up questions to refine requests before searching for the items in eBay’s inventory(ex:The bag has both brand and color ....) find the best results in the least amount of time. [20, 58]

6.2.3 Google NKG

Enables user to search for things, people or places that Google knows about (As knowledge and not string), and overflow Language ambiguous, and narrow search results that user intent. And it does not stop here, but it gives results related to user searched through semantic. [59]

6.2.4 Others

Yahoo, Microsoft's Satori[12], Facebook[12], Watson (IBM)[12], YAGO[26], cisco[60], LinkedIn[19]and big international companies such UBS Group, one of the largest financial institutions in the world[61], and big American media network [62].

[1] https://yago-knowledge.org/
[2] https://github.com/yago-naga/yago3

7 Quality assessment

The quality assessment change depends on the context what result discrepancies definitions [63] [64], we inspired [63] [64] [12]definition from [63] [64] our to it by the issues that can affect the result of the applications to do the job, also its important it vary according to need e.g. DBpedia is not insufficient for medical application but it good other purposes such as a movie archive, here we mention some common quality assessment from too many -[64] survey it- such, but there some common form the source above where they also have ddifferences between resource:

7.1 Accuracy

Accuracy refers to how entities and relations correctly represent real world phenomena from syntactic, semantic accuracy how it updated to match the real-world state in right time.

7.2 Coverage

Coverage refers to avoiding the not giving importance to the elements of the domain of application, cause bad recall from graph.

7.3 Succinctness

Succinctness refers to the inclusion only of relevant content.

7.4 Conciseness

Conciseness refers to the minimization of redundancy of entities at the schema and the data level.

8 Conclusion:

After the clarification of knowledge about the concept of knowledge graph, how can uses for reasoning and the norms that define his quality, we examine in the next chapter how to solve one of the problems that affect all of these norms which is represented in inconsistency.

Chapter 3: Consistency refine

1 Introduction

In this chapter firstly we talk about the concept of inconsistency/consistency, and how it is determined to deal with it; then introduce a framework formally that serves this purpose with all it steps and evaluate how it performs in the real world.

2 Inconsistency processing

We discover the consistency over the inconsistency where the later it is the real problem to achieving the former.

2.1 Inconsistency definition

Like the KG there no commons aforementioned definitions for inconsistency especially when it overlaps with KBs, where [65] survey ten different definitions, Minsky one of the first researchers in AI who attracted attention to the problem of inconsistent knowledge [66],some "Consistency means that a knowledge base is free of (logical/formal) contradictions with respect to particular knowledge representation and inference mechanisms.", definition we consider the flowing in the thesis "no contradictions in the data" [67].

2.2 Causes of inconsistency

KG has a flexible and scalable nature, so it is exposed to inconsistency causes, like:

1- Ontology related reasons include: either the lack of explicit constraints in the ontology specification or the discrepancy in terminology and its usage.

2- Expansion into another KG, here some from [68]:

- Epistemic conflicts: the different sources have different beliefs.

- Ontological conflicts: different sources use different terms for the same concept and/or the same term can be used by different sources for different concepts.

- KG creation and enriched from diverse sources of data that may range from plain text to structured formats.

3- Incomplete modification.

2.3 Inconsistency checking

The inconsistency checking aim to check the consistency against logical constraints or rules then give a value between 0 and 1[12]. to indicate how likely (can be refer to it by plausibility, likelihood or veracity scores) the assertion it consistent -to be true- with the KG.

There may works on inconsistency checking, first we can talk about using enriched ontology with statistical analysis Töpper et.all do for DBpedia via Inductive Logic Programming (ILP) [69], in the reason the direct using of the predefine constraint can be weak or even non-existent for some cases, and other do it with the same idea for different KGs , there also languages for constraint and rule languages such SHACL or SPARQL query templates [70] and so on, external knowledge can be encoded and applied in reasoning for assertion validation such KG.

The second way to check the consistency it by using the machine learning technique a hand with semantic embedding methods like mentioned in chapter one, from there we use observed features indicators for training, they combined with learning and prediction algorithms such supervised classification, autoencoder e.g. RDF2Vec [49], but the opposite for semantic embeddings can be learned directly by minimizing an overall loss with a score function for modeling the assertion’s likelihood algorithm such as TransE [53] but have a down side where it need high computer requirement need and also the real world KBs are noisy and sparse what make it less effective.

2.4 Inconsistency handling

The dealing with inconsistency depends on the inconsistency itself and the goal of the KB, so we can: isolate, resolve or tolerate to get a get benefit from it.

Isolate:

Don’t do anything and don't take it on concentration.

2.4.1 Tolerate

[65] comments on the opinion that see inconsistency is undesirable by: "This view is too simplistic for developing robust intelligent systems, and furthermore, it fails to use the benefits of inconsistency in intelligent activities."; [71] also they have the same opinion, where they said: "Inconsistency in information is the norm, and we should feel happy to be

able to formalize it. There are cases where q and ¬q can be perfectly acceptable together and hence need not be resolved."

2.4.2 Inconsistency resolve

Aim to repairs inconsistencies there not a lot of work on that topic [72].

Some are suggested automatic correction for a specific KBs, e.g. for DBpedia some are suggested automatic correction where correct errors during DBpedia construction [73] and Pellissier et al. [74] correct rules from the edit history of Wikidata to create a resolve model.

Ohers suggested general correction by eliminating/add (Grafting) the edges that violation constraint such [75]; but the eliminating can be it may cause loss of valuable information if the decision taken without proper deliberation [65].

Other replacing the objects or subjects with correct entities substitute by either keyword matching or a simple graph structure metric, or retrieved candidate substitutes from the Wikipedia disambiguation page (which may not exist, especially for KBs that are not based on Wikipedia) and then ranked by lexical similarity.

3 A proposed framework to refine the consistency

Here we present a framework for Jiaoyan Chen et.al [72] that can provide consistency for KGs that use 3W standards by correcting ABox property assertions ⟨s,p,o⟩ where o is a literal or an entity detected like erroneous assertion, the framework characterized by generality and does not rely on any KB meta data or external information, also can correct assertions whose objects are either erroneous entities or literals; with both semantic embeddings and observed features -with a sub-graph extracted for higher efficiency- with LP, to end with consistency checking against “soft” property constraints mined from the global KB, where exploits both deep learning and consistency reasoning for that; end the framework with a ccorrection decision making unite.

3.1 Symbols used

Here we present the symbol used on the work to be a reference:

Ɛ set of literal and/or entity assertions that have been identified as incorrect.

$\mathcal{K}$ is a KB.

T target assertion.

e' is randomly selected from E.

$\nvDash$ represents that an assertion is neither declared by the KB nor can be inferred.

$\vDash$ means an assertion is directly declared in or inferred by run can be implemented by SPARQL on the KG.

$\mathcal{K}_{\varepsilon}$ sup-graph.

3.2 General view

Basically, this framework creates a sup-graph that contains the error context-solution then apply validation on it with LP and CC (consistency checking) to get likelihood for each assertion then take the decision for each assertion by filtering the result, depend on the accuracy of consistency needed and likelihood.

The framework is mainly composed from related entity estimation, link prediction, constraint-based validation and correction decision making (fig. 11 General overview for graph).

For each a given erroneous assertion t we refer to it by target assertion where t = ⟨s,p,o⟩ , The related entity estimation identifies the entities that relevant to the object and ranked by relatedness referred to it by RE_{t}, this related entities called candidate substitutes of the original object o, the new assertions when o is replaced called candidate assertions; then the LP give a likelihood score for each $e_i \in RE_{\mathrm{t}}$ for candidate assertion t with the trained model with a subgraph from KB that represent the context of the correction; then use constraint-based validation to check the degree of consistency with KB for property cardinality and hierarchical property range; before end use correction decision making combines the results likelihood scores and consistency scores to eliminate candidate substitutes with low scores w.r.t. end user preference; in the last reports either that no suitable correction found, or recommends the most likely correction.

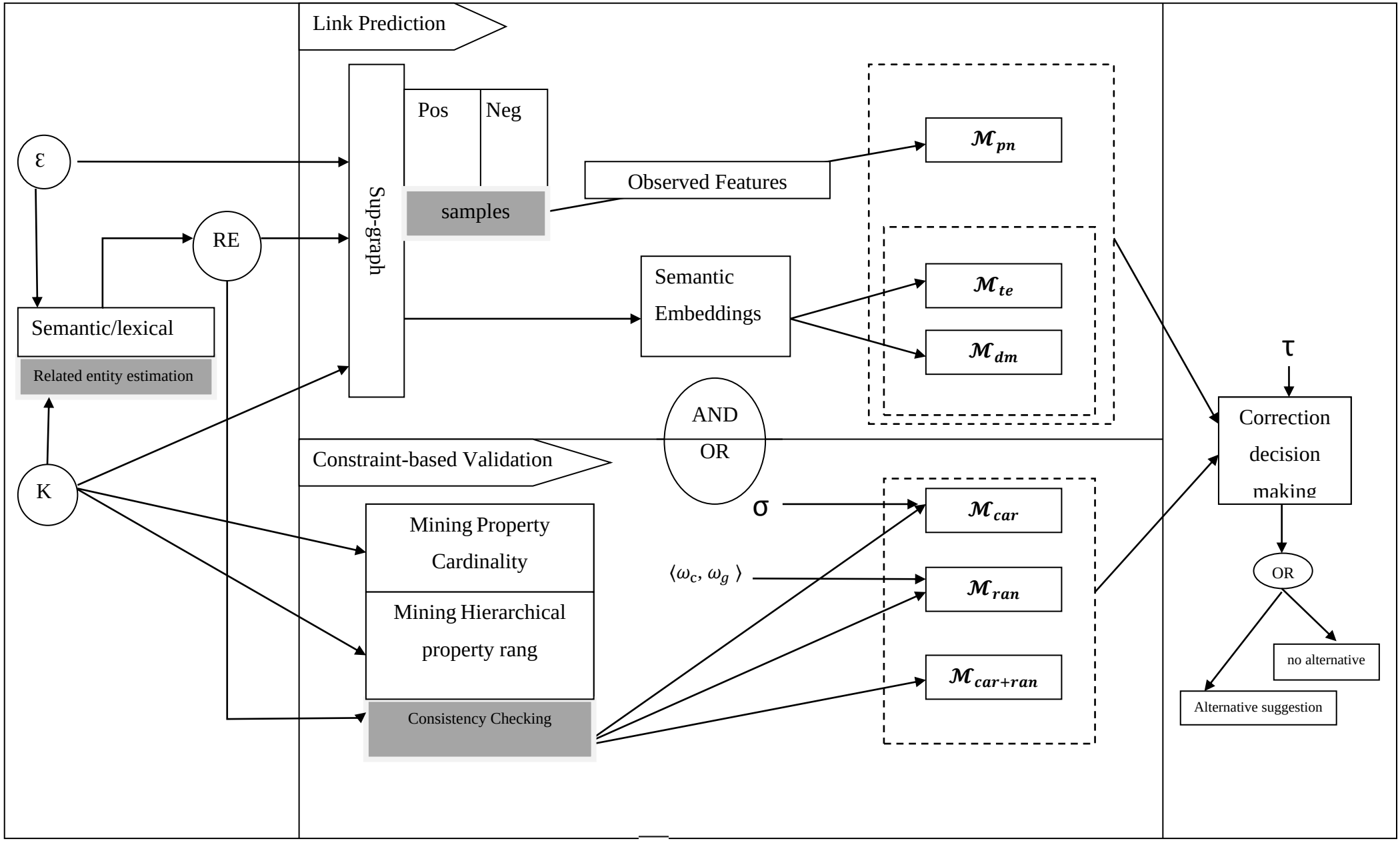

fig. 11 General overview for graph

3.3 Related entity estimation

In this step on this framework extract all the substituted candidate that can replace the object o, for each target assertion t=⟨s,p,o⟩ in Ɛ, related entity estimation takes the object o

like input if it is literal, but if o is an entity, we extract his label.

The related entity estimation returns a list RE_t containing up to k (k it that define by the end user) most related entities to the object o.

The framework supports both a lexical matching based approach and a word embedding based approach;

For the KBs with a lexical index matching directly use a lookup service based on the index, but direct lookup misses the correct entity more often, for either coming from the erroneous entity or the literal is frequently noisy and ambiguous, e.g. the DBpedia Lookup service returns no entities with the input "three gorges district" fig. 11 which refers to the entity *dbr:Three_Gorges_Reservoir_Region* that acutely exist fig. 12; to resolve this issue we use lookup* by tokenize the phrase to extract the sub-phrases for lookup with, by remove the stop words and then concatenate the tokens in their original order for sub-phrases of different lengths.

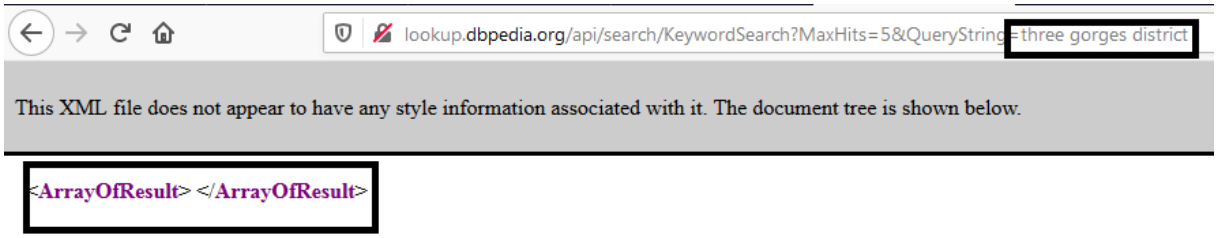

fig. 12 the DBpedia Lookup service returns no entities for "three gorges district".

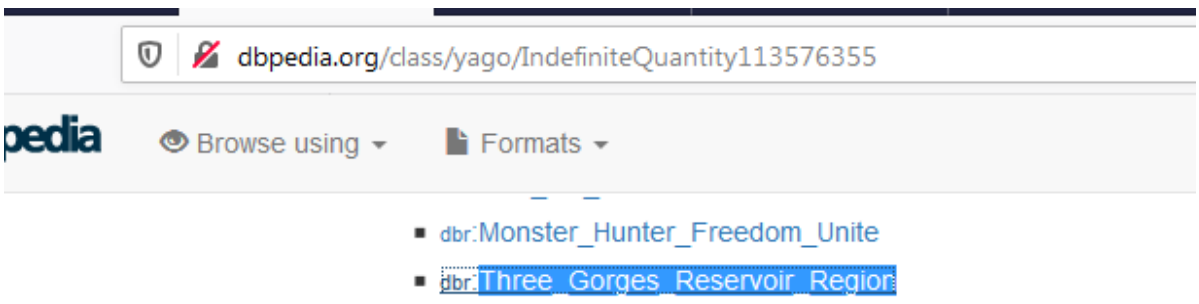

fig. 13 three gorges district in dbpedia.

The list of each lookup is ordered according to the relatedness (lexical similarity) to the original phrase, while all the lists are concatenated according to the above lookup order.

For the KBs don't support lexical index, the lexical matching based approach adopts a string similarity score named Edit Distance [76] in order to calculate the relatedness of an entity with its label.

The lexical matching with a lexical index the entity name is supported by labels in multiple languages, anchor text and different names and background description what provides more semantics what provide high value for the lookup the right entity, with the note lookup* outperforms the original Lookup.

For the word embedding we calculate the similarity of o for each entity in the KG, by using vectors.

The process composed from four step:

1 - Tokenized the phrase, and removes the stop words.

2 - Represents each token by a vector with a word embedding model (e.g., Word2Vec [52]), with ignoring the word that out of the models that trained with large amount of data.

3 - Embed the phrase by calculates the average of the vectors of all the tokens.

4 - Measure the distance-based similarity score of the two vectors by (e.g., the cosine similarity).

The word embedding technique can take the semantics of the of a word what give it assigns a high similarity score to two synonyms over the lexical matching.

The above make that example clearly, where the word "district" become noisy for lexical matching because it is not included in the label *dbp:Three_Gorges_Reservoir_Region*, but not for word embedding where “district” and “region” due to the short word vector distance between those[1].

In practice the entity issues more often caused by similarity of spelling and token composition, where the lexical similarity is high but the different.

The target of related entity estimation in the framework look for provide high recall with a not big k value, where the k set by analyzing the recall (to avoid additional noise and limit the

[1] The similarity score is determined.

size of the sub-graph),and the precision it the goal of link prediction and constraint-based validation over the candidate assertions.

3.4 Link Prediction

The framework use link prediction to predict an estimation for likelihood score to the candidate assertion ⟨s, p, e_i⟩ for each entity $e_i \in RE_t$, of the given related entities RE_t of a target assertion t = ⟨s, p, o⟩, for efficiency at start we extract the workspace that is represented in a multi-relational sub-graph, the usefulness of the space extracted it increase with the increase of the KGs volume, then train the link prediction model with it by using sampling method, different observed features and semantic embeddings.

3.4.1 Sub-graph

We start with a general description the subgraph represent the workspace from the KB with inference; T represent the all candidate assertions (in other word the replacement when actually happened -the size of the sub-graph determined by k-) we query all E element to each play o and s role alternately so still just give a likelihood for each solo assertion to be consistent (between 0 and 1), that what determine by LP and consistency score by CC.

The subgraph used to provide higher efficiency for both observed features and semantic embeddings approaches.

Hit on the sup- subgraph algorithm:

Given a KG $\mathcal{K}$ and a set of target assertions Ɛ.

The sub-graph $\mathcal{K}_\varepsilon$ is a part of $\mathcal{K}$, corresponding to Ɛ, where $\mathcal{K}_\varepsilon$= ⟨E, P, T ⟩ with E denotes entities, P denotes object properties and T denotes entity assertions.

sub-graph is calculated with three steps:

1 - Extract the seeds: get entities and properties for Ɛ, and related entities of each assertion in Ɛ.

2 - extract the neighbourhoods: directly associated assertions of each of the seed properties and entities.

3 - re-calculate to add the properties and entities form K that are in T and not exist in P and E.

Algorithm 1: Sub-graph Extraction ($\mathcal{K}$, $\mathcal{E}$, RE_t)

Input:

(1) The whole KB: $\mathcal{K}$

(2) The set of target assertions: $\mathcal{E}$

(3) The related entities of each target assertion: $RE_t,\ t \in \mathcal{E}$

Result:

The sub-graph: $\mathcal{K}_\mathcal{E} =< E, P, T >$

Begin

% Step 1: Extract the seeds

$SE = \{s| < s, p, o > \in \mathcal{E}\}$ % extract subject entities

$P = \{p| < s, p, o > \in \mathcal{E}\}$ % extract target properties

$RE = \bigcup_{t \in \mathcal{E}} RE_t$ % The union of related entities

$E = SE \cup RE$

foreach ⟨s, p, o⟩ in $\mathcal{E}$ **do**

  if o **is an** entity **then**

    $E = E \cup \{o\}$ % add object entity

% Step 2: Extract the neighbourhoods

$T = \{< s, p, o > | p \in P \wedge \mathcal{K} \vDash < s, p, o >\}$

$T = T \cup \{< s, p, o > | o \in E \wedge s \in E \wedge \mathcal{K} \vDash < s, p, o >\}$

% Step 3: Re-calculate entities and properties

$E = E \cup \{s| < s, p, o > \in T\}\{o| < s, p, o > \in T\}$

$P = P \cup \{p| < s, p, o > \in T\}$

Return $\mathcal{K}_\mathcal{E} =< E, P, T >$

Algorithm. 1 Sub-graph Extraction

3.4.2 Sampling

Positive and negative samples (assertions) are extracted from the sub-graph $\mathcal{K}_\varepsilon$=<E,P,T> to

Here extract assertions samples that are mostly to be candidate assertions compared to rest element of T.

The positive samples are composed of two parts: $T_{pos} = T_{sp} \cup T_{pr}$ where:

T_{sp} refers to assertions whose subjects and properties are among SE (i.e., those subject entities involved in Ɛ) and P respectively:

$$T_{sp} = \{< s, p, o > | s \in SE \wedge p \in P \wedge < s, p, o > \in T\}$$

T_{pr} refers to those assertions whose objects and properties are among RE (i.e., those related entities involved in Ɛ) and P respectively:

$$T_{pr} = \{< s, p, o > | p \in P \wedge o \in RE \wedge < s, p, o > \in T\}$$

T_{sp} and T_{pr} are calculated by two steps:

1) extract all the associated assertions of each property p in P from T;

2) group these assertions according to SE and RE.

This can help release the domain adaption problem the data distribution gap between the training and predicting assertions.
To make the base of samples big for more recall quality the framework suggests the flow method to create synthetic assertion, where they refer to it by T_{neg} -negative samples-.

T_{neg} are composed of two parts as well: $T_{neg} = T'_{sp} \cup T'_{pr}$ where:

T'_{sp} is constructed according to T_{sp} by replacing the object with a random entity in E.

T'_{pr} is constructed according toT_{pr} by replacing the subject with a random entity in E.

Take T_{sp} as an example, for each of its assertion ⟨s,p,o⟩, an entity e' is randomly selected from E for a synthetic assertion t' = ⟨s,p,e'⟩ such that $\mathcal{K} \nvDash t'$ and t' is added to T'_{sp}.

Note that the size of T_{pos} and T_{neg} is balanced, and $T_{neg} \cap T = \{\emptyset\}$.

3.4.3 Observed Features

The framework extract two kinds of observed features:

1) The path feature: represents potential relations that can connect the subject and object and the, where they limit the path depth to two, to reduce computation time and feature size, both of which are exponential w.r.t. the depth

2) node feature: represents the likelihood of the subject being the head of the property, and the likelihood of the object being the tail of the property.

For path feature:

First extract paths of depth one:

$$FP_{so}^{1} = \{p_0 | < \mathrm{s}, p_0, \mathrm{o} > \in \mathrm{T}\}, FP_{os}^{1} = \{p_0 | < \mathrm{o}, p_0, \mathrm{s} > \in \mathrm{T}\}$$

Secondly extract paths of depth tow, the ordered property in two directions:

$$FP_{so}^{2} = \{(p_1, p_2) | < \mathrm{s}, p_1, \mathrm{e} > \in \mathrm{T} \wedge < \mathrm{e}, p_2, \mathrm{o} > \in \mathrm{T}\}$$

$$FP_{os}^{2} = \{(p_1, p_2) | < \mathrm{o}, p_1, \mathrm{e} > \in \mathrm{T} \wedge < \mathrm{e}, p_2, \mathrm{s} > \in \mathrm{T}\}$$

Lastly, merge these paths:

$$FP = FP_{so}^{1} \cup FP_{os}^{1} \cup FP_{os}^{2} \cup FP_{so}^{2}$$

then encode them into a multi-hot vector as the path feature, denoted as f^p.

For the node feature that includes two binary variables: $f^n = [v_s, v_o]$; where:

- v_s denotes the likelihood of the subject (v_s = 1 if there exists some entity o' such that

⟨s,p,o'⟩ ∈ T and v_s = 0 otherwise).

- v_o denotes the likelihood of the object (v_o = 1 if there exists some entity s' such that ⟨s',p,o⟩ ∈ T and v_o= 0 otherwise).

Finally, calculate $f^p, f^n$and concatenate them for each sample in $T_{pos} \cup T_{neg}$, and train a link prediction model MLP:

$$\mathcal{M}_{pn} \xleftarrow[T_{pos} \cup T_{neg}]{\text{classifier e.g., MLP}} [f^p, f^n]$$

- MLP can be combined with $RDF2Vec$, and denote the model by $\mathcal{M}_{r2v}$.

3.4.4 Semantic Embeddings:

The framework uses two state-of-the-art algorithms TransE [53] and DistMult [50], for that are competitive/outperform more complex alternatives [53] [50].

In learning, a batch stochastic gradient descent algorithm is used to minimize the margin-based ranking loss:

$$L = \sum_{t \in T, t \to t'} [\gamma + g(t) - g(t')]_+$$

where γ > 0 is a hyper parameter, [·] + denotes extracting the positive part, and t' represents a negative assertion of t, generated by randomly replacing the subject or object with an entity in E.

The link prediction model of TransE and DistMult with $\mathcal{M}_{te}$or $\mathcal{M}_{dm}$ respectively.

3.5 Constraint-based validation

In this part starting with extract two kinds of soft constraints property cardinality and hierarchical property range from the KB, then use a CC algorithm to validate candidate assertions.

3.5.1 Property Cardinality

For the property p, its soft cardinality is represented by probability distribution[1]

$D_{car}^{p}(k) \in [0, 1]$, where k ≥ 1 is an integer that denotes the cardinality.

It is calculated as follows:

1 - get all the property assertions whose property is p, denoted as T(p) and all the involved subjects, denoted as S(p).

2 - count the number of the object entities associated with each subject s in S(p) and p:

$ON(s, p) = |\{o| < s, p, o > \in T(p)\}|$.

3 - find out the maximum object number: $ON_{max}^{p} = \max\{ON(s, p)| s \in S(p)\}$.

4 - $D_{car}^{p}(k) = \frac{|\{s \in S(p)|ON(s,p)=k\}|}{|S(p)|}, k = 1, \ldots, ON_{max}^{p}$

where |·| denotes the size of a set.

Note: this framework have good adopt with constraints follow CWA and UNA.

[1] The probability of cardinality k is equal to the ratio of the subjects that are associated with k different entity objects.

3.5.2 Hierarchical property rang

For the property p, its range constraint consists of: specific range and general range.

Where:

1) Specific range includes the most specific classes of its associated objects, Can be captured the direct the class instance directly denoted as C_{sp}^{p}.

2) General range: which includes ancestors of these most specific classes, denoted as C_{ge}^{p}, but excluded top classes such as *owl:Thing* being, get it with rdfs:subClassOf assertions.

Each range class c in C_{sp}^{p} (C_{ge}^{p} $resp.$) has a probability in [0, 1] that represents its supporting degree by the KB, denoted as $D_{sp}^{p}(c)$ D_{ge}^{p} $resp.$ C_{sp}^{p}, C_{ge}^{p} and the supporting degrees are calculated by the following steps:

1 - Get all the object entities that are associated with p, denoted as OE(p);

2 - Infer the specific and general classes of each entity oe in OE(p), denoted as $C_{sp}(p, oe)$ and $C_{ge}(p, oe)$ and at the same time collect C_{sp}^{p} as $\cup_{oe \in OE(p)} C_{sp}(p, oe)$

and $\cup_{ge \in OE(p)} C_{ge}(p, oe)$.

3 - Compute the supporting degrees:

$$D_{sp}^{p}(c) = \frac{|\{oe | oe \in OE(p), c \in C_{sp}(p, oe\}|}{|OE(p)|}, c \in C$$

$$D_{ge}^{p}(c) = \frac{|\{oe | oe \in OE(p), c \in C_{ge}(p, oe\}|}{|OE(p)|}, c \in C$$

The degree of each range class is the ratio of the objects that are instances of the class, as either directly declared in the ABox or inferred by *rdfs:subClassOf*.

3.5.3 Consistency Checking

The CC of the framework use the algorithm below that acts like a model.

that algorithm estimates the consistency of an assertion w.r.t. soft constraints of hierarchical property range and cardinality.

Here some bold lines for the Algorithm. 2 :

To checks the property cardinality for the given a candidate assertion t=⟨s,p,e⟩. use for checking maximum cardinality exceeding rate parameter σ ∈ (0, 1] -the higher value it means how much the p mmonopolize for the s correspond to ON^{p}_{max}- that chosen by the end user; counts the number of entity objects that are associated with s and p in the KB where t assumed to be added in that calculation, so that

$1 \leq n \leq ON^{p}_{max} + 1$.

$ON^{p}_{max} = 0$ indicates that p is most likely to used as a data property in the KB -bounces we take just entities on concentration- or even not exist; this is common in correcting literal assertions (e.g. property has_address whose objects are phrases of entity mentions but should not be replaced by entities) and other befit it to make the score of literal assertion low.

$y_{car} = 0$, when r ≥ σ and σ is set to 1.0, r ≥ σ means that p is an object property with functionality in the KB but the correction violates this constraint, from other word from too many assertions just this assertion come compete the s how associated that ON^{p}_{max}.

When ON^{p}_{max} be bigger; n the most likely will exceed; for n=1 the y_{car} indicate he will be close ON^{p}_{max}, and if n>1 being none-functional property we punish the score specially when $n > ON^{p}_{max}$ by multiplying it with a degrading factor 1 − r.

To calculate the score of consistency with considering the range property y_{ran} ($y_{ran,g}$ and $y_{ran,c}$), in the reason to check the consistency of candidate assertion we need to get the class of e then use the class of p to see the degree of the assertion support by KG. The user expresses his desire exist or by inference make a range more important the ω_g need to be higher and if he wants to need to be declared exility. The use of 1 minus to calculate how much is not supported by KB then minus 1 again to get how it supported all that to trait the t has no class so we need to return 0.

Algorithm 2: Consistency Checking ($\mathcal{M}_{ran}, \mathcal{M}_{car}$)

Input:

(1) A candidate assertion: t = <s, p, e>

(2) Property cardinality constraint: <D^{p}_{car}, ON^{p}_{max}>

(3) The maximum cardinality exceeding rate: σ ∈ (0, 1]

(4) Hierarchical property range constraint: $<D_{sp}^{p}(c), D_{ge}^{p}(c), C_{sp}^{p}, C_{ge}^{p}>$

(5) Weights of the specific range and general range: $\langle \omega_c, \omega_g \rangle$

Result:

y_{car}: score that t is consistent with the property cardinality.

y_{ran}: score that t is consistent with the property range.

Begin

% count the number of object entities

$\boldsymbol{n} = |\{\boldsymbol{o}|\mathcal{K} \vDash < s, p, o >, o \; is \; entity\} \cup \{\boldsymbol{e}\}|$**;**

$r = \frac{(n - ON_{max}^{p})}{ON_{max}^{p}}$; % calculate the exceeding rate

% no object entities are associated with p in the KB, or the cardinality exceeds the maximum by a specific rate

If $ON_{max}^{p} = 0 \;||\; r \geq \sigma$

$y_{car} = 0$;

else

If n = 1

% probability as a functional property

$y_{car} = D_{car}^{p}(k = 1)$;

Else

% probability as a none-functional property

$$y_{car} = \begin{Bmatrix} D_{car}^{p}(k > 1), & if \; r \leq 0 \\ 1 - r \,.\, D_{car}^{p}(k > 1), & else \end{Bmatrix}$$

$\boldsymbol{C}\,(\boldsymbol{e}) = \{\boldsymbol{c}|\mathcal{K} \vDash < e, \text{rdf: type}, \text{c} >\}$; % get the object's classes

% calculate the constraint score of specific and general ranges

$$\begin{cases} y_{ran,c} = 1 - \Pi_{c \in C^p_{sp} \cap C(e)} \left(1 - D^p_{ran}(c)\right), \\ y_{ran,g} = 1 - \Pi_{c \in C^p_{ge} \cap C(e)} \left(1 - D^p_{ran}(c)\right) \end{cases}$$

% calculate the overall range constraint score

$y_{ran} = \omega_c . y_{ran,c} + \omega_g . y_{ran,g}$

Return y_{ran}, y_{ran}

Algorithm. 2 Consistency Checking

At the end algorithm finally returns the property cardinality score y_{ran} and denote it model by $\mathcal{M}_{ran}$, and the property range score y_{ran} denote it model $\mathcal{M}_{car}$; those scores can be multiply or get the it average to get the final model of consistency checking denoted as $\mathcal{M}_{car+ran}$.

3.6 Correction decision making:

For the target assertion t in Ɛ, and its top-k related entities RE_t, for each entity e_i in RE_t, the correction framework do:

1 - Calculates the assertion likelihood score y_i^l with a link prediction model ($\mathcal{M}_{pn}$, $\mathcal{M}_{te}$or $\mathcal{M}_{dm}$), and the consistency score $y_i^c$with $\mathcal{M}_{car}$, $\mathcal{M}_{ran}$ or $\mathcal{M}_{car+ran}$.

2 - Separately normalizes y_i^l and y_i^c into [0, 1] according to all the predictions by the corresponding model for Ɛ.

3 - Ensembles the two scores by simple averaging: $y_i = (y_i^l + y_i^c)/2$. filters out e_i from RE_t if $y_i^l < \tau$.

Note if t is a literal assertion and its literal is exactly equal to the label of e_i then the keep e_i.

In the last stages filter entities with τ and put them into RE'_t and safe original order in RE_t, where τ is a parameter in [0, 1] that needs to be adjusted with a developing data set; at the end returns none RE'_t t is empty which means there is no entity in the KB can replace the object of t; or object substitute at the top 1 in RE'_t.

3.7 The framework in real use case:

3.7.1 Program guide:

To active the use needed pass in the agre true, execute the file:

File name	Args	functionality
DATA_generator.py	lookup_cache	get top-K entities by Loopup/Loopup*
	triple_cache_file	associated triples of each property
	object_cache_file	associated objects of each property
	entity_class_cache	classes of each entity
	class_ancestor_cache	ancestors of each class
word2vec.py	topK	get top-K entities according to word2vec
subKB.py		build the sub-KB according to the cache by cache.py
graph_feature.py	nodefeatSub	node feature
	linkfeat	path feature
constraint_mining.py	cardinality	cardinality constrain
	range	range constraint
validate_constraint.py	cardinality	property cardinality
	range	property range
	merge	merge of two consistency

3.7.2 Evaluation

For the need for large data processed and revised by specialists, we reuse the evaluation of the paper because is more realistic. The image/tables from them:

DBP-Lit ($\mathcal{E}$): target assertion for Dbpedia.

MED-Ent ($\mathcal{E}$): a set of assertions with erroneous entity objects that have been discovered and collected during deployment of the KB in enterprise products.

GT ($\mathcal{K}$): Ground truths are carefully checked using DBpedia, Wikipedia, and multiple external resources.

Compared different methods and settings for related entity estimation

result in fig. 14:

1 - The lexical matching based methods (Lookup, Lookup$*$ and Edit Distance) have much higher recall than Word2Vec, on both DBP-Lit and MED-Ent, because DBpedia Lookup service takes not only the entity label but also the anchor text into consideration It provided a

strong semantic that result good recalling; and other side the MED-Ent erroneous objects are often caused by lexical confusion, such as misspelling and misusing of an entity with similar tokens.

2 - Lookup∗ proposed by the framework outperforms the original Lookup.

3 - lexical matching using either Lookup (for DBP-Lit) or Edit Distance (for MED-Ent) more correction rate and accuracy than Word2Vec

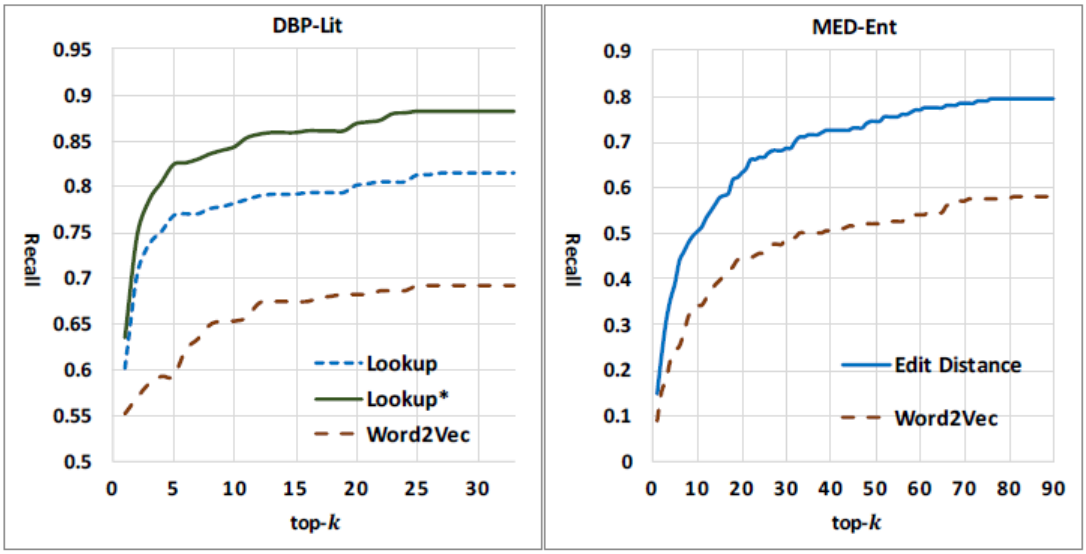

fig. 14 The recall of Entity GTs by top-k related entities.

Link Prediction Models impact:

result in Table 3:

- Node+Path or each alone outperformance over RDF2Vec for DBP-Lit.

- Node+Path performs better than TransE and Dist- Mult for DBP-Lit, while for MED-Ent, TransE and DistMult outperforms Node+Path.

Table 3 Link prediction results based on the whole KB, and their outperformance

Cases	MRR	Hits@1	Hits@5
TransE (MED-Ent)	0.713 (-.031)	0.608 (-.044)	0.834 (-.028)
DistMult (MED-Ent)	0.766 (+.014)	0.721 (+.027)	0.822 (+.016)
Node+Path (DBP-Lit)	0.504 (-.082)	0.384 (-.077)	0.611 (-.105)

4 Conclusion:

We have seen in this chapter how the inconsistency is a dilemma, that it is not easy to deal with, and that it is not always a problem where can be used in some cases. Then we see the possible ways to deal with it. Finally, we presented a solid framework with evaluation for its usefulness degree in actual life.

Chapter 4: Conclusion

We have treated in this work the problem of the consistency of KG. We have reviewed the basic knowledge to access KG consistency. With this in mind, we have succeeded in:

1 - Talked about the most data models that impact on practice and the position of KG from them.

2 - Clarified the concept of the KG with the mention the ado about it, in particularly the overlapped with other terms and academic researchers' efforts to get accurate definition, with presenting our own vision.

3 - Shows the necessary enrichment to the graph data model with a schema, identifiers and context in order to construct a knowledge graph.

4 - Talked about how the Knowledge graph can be intelligent via inductive and deductive reasoning for (semi-)automatically dealing with task because at last the correction (or any deal on general) should be automatic.

5 - Give over view for KGs especially that are important for developers (Dbpedia, YAGO).

6 - Talked about quality assessment and talk about the instability of this concept.

Subsequently, we tackled the issue of the consistency of KGs, by:

1 - Present the problem of the inconsistency where we mention some different definitions and mention also to the dispute about that term.

2 - Present inconsistency causes and the various ways to detect and deal with.

3 - Highlight correction methods exist and mention some works in this field.

4 - Present a framework and explanation it's ideas that was mysterious.

In overall conclusion, we find that this topic was fun, informative and modern, we learn a lot from it and improve our programing skills and dealing with scientific articles; however, with the higher value the more difficult the road, so we would like to mention some difficulties that we encountered:

1 - The lack of resources about these concepts on the internet and books, so we had to search in specialized research papers, each of which That takes time and effort. This may be due to the novelty of the field.

2 - We had to explore another parallel area which is deep learning and both artificial intelligence and data quality/models slightly lighter. Given the hardness of both areas, this exploration was not without cost in terms of time and effort.

3 - The lack of consensus in the community regarding the two key concept in this issue, namely: Knowledge Graph and Consistency-, and also quality assessment.

Finally, we consider the following perspectives as open questions to be addressed in the near future:

On the assertion correction that follow the perspectives of framework previously submitted:

- Improve recall of related entity, e.g. via incorporating external resources like Wikipedia disambiguation pages or domain knowledge about the errors.

- Improve integration of link prediction and constraint-based validation.

And for the framework perspective or any new perspective there the need into improve the using self-supervised methods and graph embedding to dealing with noisy and sparse knowledge domain in the real world, this will result a huge leap for AI agent to deal with inconsistency to resolve or provide value from.

References

[1] Angles, R. and Gutierrez, C. Survey of graph database models, 40, 1 %J ACM Comput. Surv. (2008), Article 1.
[2] Ilyukha, V. Differences Between Relational and Non-Relational Database (
[3] Williams, P. The NoSQL Movement What is it? (October 11 2012).
[4] Codd, E. F. Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks (1969).
[5] Abiteboul, S. Querying Semi-Structured Data (1997.).
[6] Klarman, S. Modelling Data with Hypergraphs A closer look at Grakn Labs hypergraph data model (Apr 19, 2017).
[7] Sasaki, B. M. Graph Databases for Beginners: ACID vs. BASE Explained (2018).
[8] Xu, C. and Zhou, A. Quality-aware Scheduling for Key-value Data Stores (2015.
[9] Foote, K. D. Understanding Key-Value Databases (2020).
[10] aws.amazon.com What Is a Document Database? (2020).
[11] Blaschka, T. It's Not Just Hype Graph is Transforming Businesses (26 Aug 2020).
[12] HOGAN, A., BLOMQVIST, E., COCHEZ, M., D'AMATO, C., MELO, G. D., GUTIERREZ, C., GAYO, J. E. L., KIRRANE, S., NEUMAIER, S., POLLERES, A., NAVIGLI, R., NGOMO, A.-C. N., RASHID, S. M., RULA, A., SCHMELZEISEN, L., SEQUEDA, J., STAAB, S. and ZIMMERMANN, A. Knowledge Graphs (2020).
[13] Bryce Merkl Sasaki Graph Databases for Beginners: Why Connected Data Matters (jul 17 2018).
[14] VUKOTIC, A., WATT, N., ABEDRABBO, T., FOX, D. and PARTNER, J. Neo4j in Action (2015).
[15] Rusu, F. and Huang, Z. In-Depth Benchmarking of Graph Database Systems with the Linked Data Benchmark Council (LDBC) Social Network Benchmark (SNB) (July 2019).
[16] Ehrlinger, L. and Wöß, W. Towards a Definition of Knowledge Graphs (2016).
[17] Bergman, M. K. A COMMON SENSE VIEW OF KNOWLEDGE GRAPHS (July 1, 2019).
[18] Devarajan, D. Happy BirthdayWatson Discovery (2017).
[19] Qi He, B.-C. C. and Agarwal., D. Building The LinkedIn Knowledge Graph. LinkedIn Blog. (2016).
[20] Pittman, R., Srivastava, A., Hewavitharana, S., Kale, A. and Mansour, S. Cracking the Code on Conversational Commerce (Apr 6 2017).
[21] Schneider, E. W. Course Modularization Applied [microform] : The Interface System and Its Implications For Sequence Control and Data Analysis (april 1972).
[22] Biggs, N., Lloyd, E. and Wilson, R. Graph Theory, 1736-1936 (1986).
[23] Hayes, P. J. What is a Knowledge Graph? CORRECTION (15 Jun 2019).
[24] Nurdiati, S. i. and Hoede, C. 25 YEARS DEVELOPMENT OF KNOWLEDGE GRAPH THEORY:THE RESULTS AND THE CHALLENGE (
[25] Pujara, J., Miao, H., Getoor, L. and Cohen, W. Knowledge Graph Identification (2013).
[26] Färber, M., Ell, B., Menne, C., Rettinger, A. and Bartscherer, F. Linked Data Quality of DBpedia, Freebase,OpenCyc, Wikidata, and YAGO. *Semantic Web* (2016)).
[27] Blumauer, A. From Taxonomies over Ontologies to Knowledge Graphs (July 15 2014).
[28] Stichbury, J. WTF is a knowledge graph? (2019).
[29] Assom, L. Answer1 How do you build and maintain a knowledge graph, i.e, what is the suggested path to extract, model, generate a database and visualize a graph that is similar in concept to Google knowledge graph? (NOVEMBER 2018).
[30] Noy, N., Gao, Y., Jain, A. K., Narayanan, A., Patterson, A. and Taylor, J. J. C. o. t. A. Industry-scale knowledge graphs, 62 (2019), 36 - 43.
[31] Barrasa, J. RDF Triple Stores vs. Labeled Property Graphs: What's the Difference? (18 aug 2017).
[32] Cyganiak, R., Wood, D. and Lanthaler, M. RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014 (2014).
[33] Gilbert, E. Triplestores 101: Storing Data for Efficient Inferencing (sep 2016).
[34] Carothers, G. and Lex Machina N-Quads - A line-based syntax for RDF datasets (2014).

[35] Cyganiak, R., Wood, D. and Lanthaler, M. RDF 1.1 Concepts and Abstract Syntax (25 February 2014).
[36] Sasaki, B. M. Graph Databases for Beginners: Other Graph Technologies (Dec 11 2018).
[37] Robinson, I., Webber, J. and Eifrem, E. Graph Databases NEW OPPORTUNITIES FOR CONNECTED DATA (2015).
[38] Martinez-Rodriguez, J., Hogan, A. and Lopez-Arevalo, I. Information extraction meets the Semantic Web: A survey. *Semantic Web*, 11 (10/26 2018), 1-81.
[39] Arenas, M., Bertails, A., Prud'hommeaux, E. and Sequeda, J. A Direct Mapping of Relational Data to RDF (2012).
[40] Bischof, S., Decker, S., Krennwallner, T., Lopes, N. and Polleres, A. Mapping between RDF and XML with XSPARQL (2012).
[41] Koide, S. Knowledge Graph Considered Harmful for Ontology (
[42] Sequeda, J. Introduction to: Open World Assumption vs Closed World Assumption (November 30 2012).
[43] Peterson, D., Gao, S. S., Malhotra, A., Sperberg-McQueen, C. M. and Thompson, H. S. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes (2012).
[44] Pascal Hitzler, M. K., Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language (2012).
[45] Brickley, D. and Guha, R. V. RDF Schema 1.1 (2014).
[46] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F. and Rudolph., S. OWL 2 Web Ontology Language Primer (Second Edition) (2012).
[47] Prud'hommeaux, E., Jose Emilio Labra Gayo and Solbrig, H. Shape Expressions: An RDF Validation and Transformation Language (2014).
[48] Knublauch, H. and Kontokostas, D. Shapes Constraint Language (SHACL) (2017).
[49] Ristoski, P. and Paulheim, H. *RDF2Vec: RDF Graph Embeddings for Data Mining*, 2016.
[50] Yang, B., Yih, W.-t., He, X., Gao, J. and Deng, l. Embedding Entities and Relations for Learning and Inference in Knowledge Bases (12/19 2014).
[51] Ji, G., He, S., Xu, L., Liu, K. and Zhao, J. *Knowledge Graph Embedding via Dynamic Mapping Matrix*. City, 2015.
[52] Mikolov, T., Chen, K., Corrado, G. s. and Dean, J. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR*, 2013 (01/16 2013).
[53] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J. and Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data (2013), 2787--2795.
[54] Wang, Q., Mao, Z., Wang, B. and Guo, L. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29, 12 (2017), 2724-2743.
[55] Han, X., Cao, S., Lyu, X., Lin, Y., Liu, Z., Sun, M. and Li, J. *OpenKE: An Open Toolkit for Knowledge Embedding*, 2018.
[56] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S. and Zhang, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (08/24 2014).
[57] Krishnan, A. Making search easier How Amazon's Product Graph is helping customers find products more easily. (August 17 2018).
[58] Neo4j Neo4j Powers Intelligent Commerce for eBay App on Google Assistant (2019).
[59] Singhal, A. Introducing the Knowledge Graph: things, not strings (May 16 2012).
[60] Neo4j Cisco Systems Real-Time Graph Analysis of Documents Saves Company Over 4 Million Employee Hours (2019).
[61] Neo4j UBS Data Lineage Tool Improves Risk Management, Drives Compliance (2019).
[62] Neo4j NBC News Analyzes Hundreds of Thousands of Russian Troll Tweets Using Neo4j (2019).
[63] Batini, C. and Scannapieco, M. *Data Quality: Concepts, Methodologies and Techniques*, 2006.
[64] Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J. and Auer, S. Quality assessment for Linked Data: A Survey. *Semantic Web*, 7 (03/17 2015), 63-93.
[65] Hunter, A. and Konieczny, S. *Approaches to Measuring Inconsistent Information*, 2005.

[66] Burgin, M. and de Vey Mestdagh, C. N. J. Consistent structuring of inconsistent knowledge. *Journal of Intelligent Information Systems*, 45 (08/24 2015), 5-28.
[67] Hogan, A., Harth, A., Passant, A., Decker, S. and Polleres, A. *Weaving the Pedantic Web*, 2010.
[68] Hunter, A. and Summerton, R. A knowledge-based approach to merging information. *Knowledge-Based Systems*, 19 (12/01 2006), 647-674.
[69] Töpper, G., Knuth, M. and Sack, H. DBpedia ontology enrichment for inconsistency detection. *ACM International Conference Proceeding Series* (09/05 2012).
[70] Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R. and Zaveri, A. *Test-driven Evaluation of Linked Data Quality*, 2014.
[71] Gabbay, D. and Hunter, A. *Making inconsistency respectable: A logical framework for inconsistency in reasoning, part I — A position paper*. City, 2006.
[72] Chen, J., Chen, X., Horrocks, I., Jimenez-Ruiz, E. and Myklebus, E. B. Correcting Knowledge Base Assertions (2020).
[73] Dimou, A., Kontokostas, D., Freudenberg, M., Verborgh, R., Lehmann, J., Mannens, E., Hellmann, S. and Walle, R. *Assessing and Refining Mappings to RDF to Improve Dataset Quality*. City, 2015.
[74] Tanon, T. P., Bourgaux, C. and Suchanek, F. Learning How to Correct a Knowledge Base from the Edit History. In *Proceedings of the The World Wide Web Conference* (San Francisco, CA, USA, 2019). Association for Computing Machinery, [insert City of Publication],[insert 2019 of Publication].
[75] de Melo, G. Not quite the same: Identity constraints for the Web of Linked Data. *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013* (01/01 2013), 1092-1098.
[76] Navarro, G. A guided tour to approximate string matching, 33, 1 %J ACM Comput. Surv. (2001), 31–88.