Knowledge graph extensions: consistency, immutability, reliability, and context

Savaş Takan Corresp. 1

1 artificial intelligence and data engineering, ankara üniversity, ankara, turkey

Corresponding Author: Savaş Takan
Email address: savastakan@gmail.com

Although the knowledge graph is a very convenient tool for storing knowledge in artificial intelligence, it has several areas for improvement. These shortcomings can be summarized as the problem of automatically updating all the information that affects it when a piece of information changes, ambiguity, the inability to sort information, the inability to keep some information immutable, and the inability to make a quick comparison between information. In our work, we integrate reliability, consistency, immutability, and context mechanisms into the knowledge graph to contribute to solving these problems. In our work, these plugins are kept separate from the knowledge graph so that the functionality of the structures is not impaired. The hash mechanism is used in the design of these plugins. This paper describes the plugins in detail and then proves their functionality by testing them with a sample application. The plugins are expected to contribute to the knowledge graph optimization literature and the development of artificial intelligence software that utilizes knowledge graphs.

Knowledge Graph Extensions: Consistency, immutability, Reliability, and Context

Savaş Takan[1]

[1]**Artificial Intelligence and Data Engineering, Ankara University, Ankara, Turkey**

Corresponding author:
First Author[1]

Email address: stakan@ankara.edu.tr

ABSTRACT

Although the knowledge graph is a very convenient tool for storing knowledge in artificial intelligence, it has several areas for improvement. These shortcomings can be summarized as the problem of automatically updating all the information that affects it when a piece of information changes, ambiguity, the inability to sort information, the inability to keep some information immutable, and the inability to make a quick comparison between information. In our work, we integrate reliability, consistency, immutability, and context mechanisms into the knowledge graph to contribute to solving these problems. In our work, these plugins are kept separate from the knowledge graph so that the functionality of the structures is not impaired. The hash mechanism is used in the design of these plugins. This paper describes the plugins in detail and then proves their functionality by testing them with a sample application. The plugins are expected to contribute to the knowledge graph optimization literature and the development of artificial intelligence software that utilizes knowledge graphs.

INTRODUCTION

Since time immemorial, acquiring, storing, and managing knowledge has been one of the main goals of humanity. Today, thanks to developing technologies, information is multiplying very rapidly. Therefore, it becomes difficult to process, infer and use information. Most of these problems are related to how knowledge is represented. One of the most widely used methods of knowledge representation is the knowledge graph.

Knowledge graphs have emerged as an essential area in artificial intelligence in the last decade (Rajabi and Etminani, 2022). A knowledge graph can simply be a directed, labeled, multi-relational graph with some form of semantics (Kejriwal, 2022). A knowledge graph, also known as a semantic network, refers to a graphical representation of real-world entities and relationships; objects, events, situations, or concepts and the relationship between them. A knowledge graph is essential for storing and making inferences from it.

In recent years, knowledge graphs have been widely applied for different purposes in various domains. In parallel, there have been studies on their integration with various domains. These include the creation of semantic knowledge graphs for news production, distribution, and consumption in digital news platforms (Opdahl et al., 2022), the integration of heterogeneous knowledge sources in the creation of large knowledge graphs and Artificial Intelligence (AI) systems to be more explainable and interpretable (Rajabi and Etminani, 2022), the application of knowledge graphs in manufacturing and production, reasoning technologies in knowledge graphs (Chen et al., 2020b), the Semantic Web (Ryen et al., 2022), applying machine learning, rule-based learning and natural language processing tools and approaches (Verma et al., 2022), and how statistical models can be trained on large knowledge graphs and used to predict new facts about the world (Nickel et al., 2016).

Although the knowledge graph is a very convenient tool for storing knowledge in the field of artificial intelligence, there are some essential requirements and shortcomings for it, no matter which field it is used in. These shortcomings can be summarized as the problem of automatically updating all the information that affects a piece of knowledge when it changes, the inability to sort information, the inability to keep some information immutable, and the inability to make a quick comparison between information

(Kejriwal, 2022; Troussas and Krouska, 2022; Noy et al., 2019). In our work, reliability, consistency, immutability, and context mechanisms are integrated into the knowledge graph to contribute to solving these problems.

The knowledge graph must be always consistent (Mu, 2015). This consistency may be lost if any information changes. To restore coherence, all the information connected to the changed information must change. This is because a change in the elements that support a piece of knowledge, with a chain effect, calls into question the reality of all the elements supported by that knowledge. The time factor is vital to ensure consistency in the knowledge change (Terenziani, 2000). In addition, consistently keeping knowledge helps to reduce complexity (Liberatore and Schaerf, 2001). The classical knowledge structure can find changing knowledge by cause-effect and inference. However, since such methods do not have stamping and tracking, they are complex and can lead to overlooking information that needs to change. Moreover, if this inference is global, it will have performance problems, and if it is local, it will return conflicting information because it cannot capture change. At the same time, there are severe performance penalties when erroneous information is removed, new information is added, or existing information is modified.

Another requirement for the knowledge graph is to ensure the ordering of information (Porebski, 2022). We have integrated a reliability mechanism into the knowledge graph to fulfill this requirement. Accordingly, the more reliable elements supporting a piece of information, the more reliable that information is considered to be. In the opposite case, the information in question is interpreted as doubtful. Thus, ranking between information becomes possible.

Another requirement in the knowledge graph is the comparison of two pieces of information (Wu et al., 2021; Jabla et al., 2022). It is vital that this comparison can be made very quickly. In our work, we integrate a hashing mechanism called context into the knowledge graph, which allows us to determine the identity of two pieces of information in O(1) time. Context allows the disambiguation of a piece of knowledge by looking at its contexts. For example, Jaguar refers to both an animal and a programming language. The ambiguity about which of these is expressed in a knowledge graph can be resolved by comparing its constituent knowledge, thanks to the context plugin we have developed.

Another vital element of the knowledge graph is that knowledge can be immutable (Cano-Benito et al., 2021; Besançon et al., 2022). For example, while the people who buy or read a book can change, the book's title and the author must be immutable. In other words, some elements can change in knowledge, and elements do not change.

The proposed plugin ensures consistency by marking the information as soon as it changes and updating the associated information to run in the background at any time. To ensure that the information is immutable, a structure has been created to store immutable and mutable data. Regarding the reliability of the information, an information hierarchy has been developed in the system. Regarding context, the summarization function provides unique hash values for existing contexts. Thus, when there is a match between different contexts of two pieces of information, it can be quickly recognized that they have the same context.

In the study, the research on the subject is given respect, and then the methodology of the proposed plugins is explained. Then, the plugins are explained in detail, and their advantages and disadvantages are presented.

1 RELATED WORK

There is a vast literature on the knowledge graph. There are primarily many review papers on the topic (Chen et al., 2021; Cambria et al., 2021; Chen et al., 2020a; Issa et al., 2021; Dai et al., 2020).

Several studies in the literature focus on consistency in the knowledge graph. In one of the most influential early studies, two new complementary features on constraints in a network were developed (van Beek and Dechter, 1997). The authors suggest that these features can be used to decide whether it would be helpful to pre-process the network before a callback search. In a different study, tools for consistency checking were found to provide an opportunity to reduce minor inconsistencies in the Gene Ontology (GO), and redundancies in its representation (Yeh et al., 2003). Another study presented a general, consistency-based framework for expressing belief change (Delgrande and Schaub, 2003). With this framework, other belief change operations, such as updating and deleting, can also be expressed. In another study, a measurement parameter was developed to quantify the amount of inconsistency in probabilistic knowledge bases (Muiño, 2011). In the study, inconsistency was measured by considering the minimum adjustments in the degrees of certainty of statements (i.e., probabilities in this paper) necessary to make the knowledge base consistent. In a different study, Mu proposed a measure for the degree of responsibility of each formula in a knowledge base for the inconsistency of that base (Mu, 2015). This measure is given in terms of the minimum, inconsistent subsets of a knowledge base.

More recent literature on the topic includes studies that address the central problem of the computational complexity of consistency checking (Grant et al., 2018), as well as a graph-based approach to measuring inconsistency for a knowledge base (Mu, 2018) better to understand the nature of inconsistency in a knowledge base. Another recent study starts from the challenges of the belief revision process (Bello López and De Ita Luna, 2021). Accordingly, one of the most critical problems is how to represent the knowledge base K to be considered and how to add new information. In this paper, an algorithmic proposal is developed to determine when (K E (K *)) is inconsistent.

Besides consistency, context is central to many modern safety and security-critical applications. In a different study, the phrase similarity of human comments was determined using four different methods, including item matching, linguistic collocation approaches, and wordnet semantic network distance (Stock and Yousaf, 2018). The method that incorporates context is said to be the most successful of the four methods tested, selecting the same geometric configuration as human respondents in 69% of cases. In another study on the context in knowledge graphs, a formal approach to achieve contextual reasoning was developed based on the lack of formal integration of knowledge and context in existing context-aware systems (Alsaig et al., 2020).

In the literature, there are many studies on the ordering of nodes in graph theory (Sciriha and da Fonseca, 2012; Nirmala and Nadarajan, 2022; Huang et al., 2021; Christoforou et al., 2021). However, as far as we know, there is no research on ordering in the knowledge graph. At the same time, although the issue of immutability in data structures has been frequently studied (Chowdhury et al., 2018; Ozdayi et al., 2020; Stančić and Bralić, 2021; Balakrishnan et al., 2019), there is no research on immutability in knowledge graphs. In addition, although several studies focused on reliability and ranking in the knowledge graph (Seo et al., 2020; Yang et al., 2022; Jiang et al., 2022), these studies are not directly related to the topic of our article. Similarly, a limited number of studies focused on hashing in the knowledge graph (Khan et al., 2023; Wang et al., 2020). Still, the existing studies in the literature are not related to the plugins we developed in our article.

As can be seen, studies on the knowledge graph in the literature have covered a wide range of topics. Studies have generally focused on integrating the knowledge graph into other domains. In the literature, studies focusing on consistency in the knowledge graph have generally developed complex solutions. In the limited number of context-oriented studies in the literature, application-based solutions have been developed without any change in the structure of the knowledge graph. Our study differs from the existing studies in the literature that focus on consistency and context in the knowledge graph by providing these extensions with hashing technology. This is because no studies in the existing literature integrate invariance, consistency, reliability, and context into the knowledge graph using hashing technology. The main contribution of our work to the literature is to show that four different properties can be integrated into the knowledge graph with a simple mechanism (Hashing). In this respect, our work is expected to contribute to the literature on better representation of knowledge and the solutions created, and to the development of artificial intelligence software using knowledge graphs.

2 MATERIAL AND METHODS

In our work, consistency, context, reliability, and immutability mechanisms are integrated into the knowledge graph to perform the operations of where existing knowledge comes from, by whom it is supported, the rate of support, ranking, and whether it is a modifiable or immutable and automatic update. Unlike the literature, these extensions were developed using the hashing mechanism. This is because Hashing technology, a straightforward mechanism, offers the possibility to provide four different properties quickly. In our work, a " Knowledge " model provides consistency, immutability, reliability, and context extensions to the knowledge graph.

Thanks to the hashing mechanism, it is possible to check whether the relationships and data in the information have changed. Relationships that are checked whether they change are called constant relationships, and relationships that are not checked are called variable relationships. On the other hand, data is constantly checked and therefore considered constant in the knowledge graph. Figure 1 shows the general features of the knowledge graph we developed.

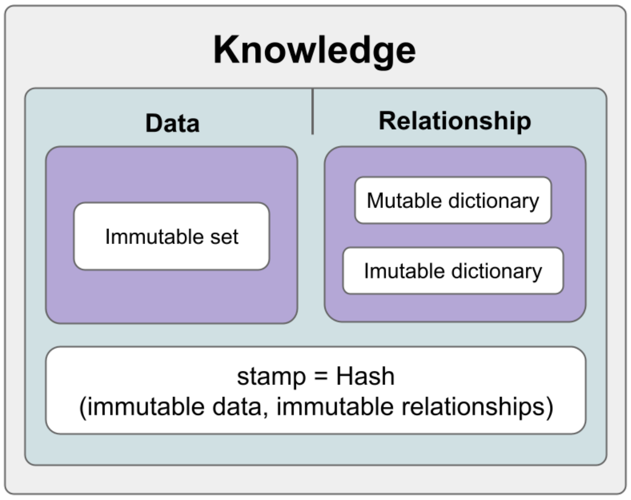

Figure 1. The general scope of knowledge

The immutability control in the knowledge graph is provided by the hash mechanism. Here, a hash is a hash of immutable relations and data. The hash is calculated and added to the hash set at any time. In the information structure we have developed, this is done with the lock() function. Then, a new hash value is calculated and compared with the old hash values in the hash set to check if there are any changes in the information. After that, if there is a change in the relations or data of the information, a different hash value will be output, so it can be automatically determined whether the information has changed and, if so, its position. If the results are equal, the structure has not been changed; if the results are not equal, it means that the structure has been changed. This is done with the isLock() function. The general structure of the lock and islock state of the information is shown in Figure 2.

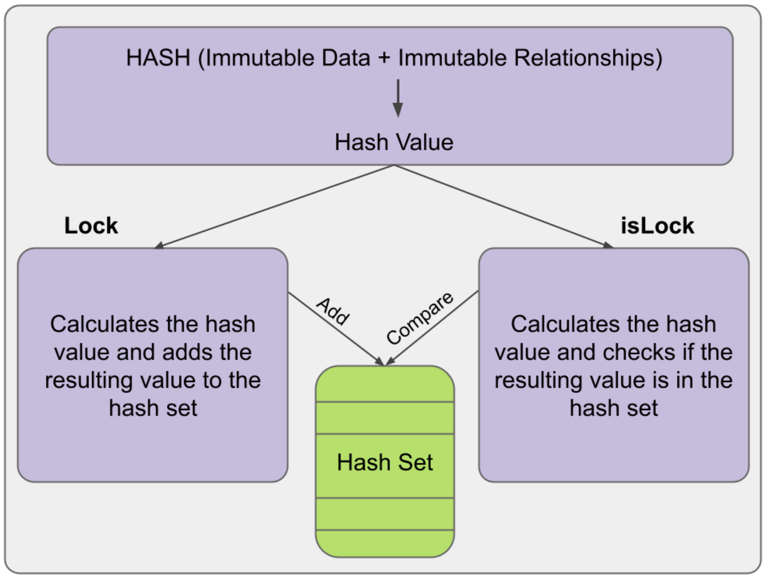
HASH (Immutable Data + Immutable Relationships)
Hash Value
Lock
isLock
Calculates the hash value and adds the resulting value to the hash set
Add
Compare
Calculates the hash value and checks if the resulting value is in the hash set
Hash Set

Figure 2. Lock and Unlock states

An example hash-finding formula is as follows. Here, the value i indicates how many of the n fixed data are expressed. The value j indicates how many of the m fixed relations are expressed.

Calculating the information hash value:

$$hash(\sum_{i=0}^{n} Immutable\,data_i + \sum_{j=0}^{m} Immutable\,relationship_j) \quad (1)$$

When a piece of information is deleted, modified, created, or added, the information that depends on it is recalculated and updated by Algorithm 1 to maintain consistency. In parallel, Algorithm 1 can detect if there has been a change in the knowledge graph. Algorithm 1 ensures changes are propagated to all low-complexity points in the knowledge graph. In addition, the same algorithm can also be used to find where the changes in the knowledge graph have occurred. In the algorithm, updates are performed on the invariant relations in the knowledge graph. In other words, variable relations are not taken into account. This algorithm was developed using depth-first search, dynamic programming, and topological ranking.

Algorithm 1 The algorithm below updates all affected nodes, egdes, whenever there is a change in any node.

```
stack ← [$startEdge$]
$visited$ ← []
while $stack$ do
    for all neighbor_edge ∈ graph.edges(edge[1]) do
        if neighbor_edge ∉ visited then
            visited.append(neighbor_edge)
            if edge ∉ parent[neighbor_edge] then
                parent[neighbor_edge].append(edge)
                cost ← weight_cost[edge] + graph[neighbor_edge]['weight']
                weight_cost[neighbor_edge] += cost
            end if
        end if
    end for
end while
```

Since the knowledge graph is a cyclic graph with multiple transitions, nodes and edges are swapped to traverse all transitions. Thus, all edges can be traversed. In this way, the whole system is traversed with O(E) complexity. As a result, the whole system can be updated with linear complexity. In the knowledge graph, the update can be determined according to the depth parameter given by the user. Thus, the user can determine how many depth units can be updated.

3 LIMITATIONS

To physically test the model we developed, four STM32 and Lora Modules were used, and tests for readability, storability, and manipulation of data were performed. As a result, it was determined that the system could physically operate without problems. However, for financial reasons, more comprehensive and holistic tests could not be carried out at this stage, and it was impossible to test the model we developed on large systems. However, such a test in our work is considered necessary and valuable. For this reason, more comprehensive applications are planned to be realized by providing the necessary resources.

Immutability, reliability, consistency, and contextualization are not elements that can be easily tested. For this reason, in our study, we have tried to prove the applicability of these elements through example scenarios. In future studies, it is thought that running it on real scenarios would be helpful.

4 KNOWLEDGE GRAPH EXTENSIONS

This section describes each plugin we developed for the knowledge graph and proves their functionality by testing them with example scenarios. Thus, it is shown that our knowledge graph plugins can be used in a wide variety of software processes.

4.1 Immutability

To illustrate the uncontrolled relationship, five different pieces of information are constructed below. A has an outward relationship with C and B through k and j. B, C, j, k have no relationship at this stage. The JSON format used to generate these five pieces of information is as follows:

k: {{'do'}, Null, Null, Null, Null}

j: {{'sell'}, Null, Null, Null, Null}

B: {{'Customer 1'}, Null, Null, Null, Null}

C: {{'Image 1'}, Null, Null, Null, Null}

A: {{'Painter'}, {k: C}, {j: B}, Null}

In the phase shown above, the lock function has not been executed. Therefore, no immutability mechanism has been activated, meaning that the hash values will be shown as Null. In the JSON representation above, there are four values. The first is the checked data. The second is the checked relations, the third is the unchecked relations, and the fourth is the hash value. By calling the lock function of A with the following command, the system is locked and thus made unalterable. The command to call the lock function of A is as follows:

```
A.lock()
```

After the Lock function is applied, the JSON format view of the structure follows. The point to note here is that hash values are entered. Since C and k information is dependent on A, when A information is made immutable, this information also becomes immutable. On the other hand, since j and B are not checked (they have a variable relationship), they are not fixed, and the hash value remains Null. This can be easily seen in Formula 1. Furthermore, with the calculation in Formula 1 and Algorithm 1, A can determine whether the information in k and C has changed, and if so, which information has changed.

k: {{'do'}, Null, Null, Null, {3e4d}}

j: {{'sell'}, Null, Null, Null, Null}

B: {{'Customer 1'}, Null, Null, Null, Null}

C: {{'Image 1'}, Null, Null, Null, {13h}}

A: {{'Painter'}, {k: C}, {j: B}, {12ew}}

At any time, a new piece of information can be added linked to the variable relation, and the hash value will not change even if the lock function is executed. This provides design flexibility. Because some relations are fixed while others are variable. For example, the painter of a work of art is fixed, while the customers who buy this work of art are variable. It isn't easy to create this structure in the knowledge graph. Below, it is shown in JSON format that hash values do not change even if the relations we do not control (variable) change:

k: {{'do'}, Null, Null, Null, {3e4d}}

j: {{'sell'}, Null, Null, Null, Null}

B: {{'Customer 2'}, Null, Null, Null, Null}

C: {{'Image 1'}, Null, Null, Null, {13h}}

A: {{'Painter'}, {k: C}, {j: B}, {12ew}}

Suppose new information is added to the immutable relation on demand. In that case, the hash values are reconstructed, and when these new values are compared with the old ones, it will be seen that the newly created hash values are different from the old ones. The point we would like to draw attention to here is that the hash value of the A information will change when a new D information is added to the above example. This is shown below in JSON format:

k: {{'do'}, Null, Null, Null, {3e4d}}

j: {{'sell'}, Null, Null, Null, Null}

B: {{'Customer 2'}, Null, Null, Null, Null}

C: {{'Image 1'}, Null, Null, Null, {13h}}

D: {{'Picture 1'}, Null, Null, Null, {12de}}

A: {{'Painter'}, {k: C, k: D}, {j: B}, {1saw}}

Diversity for uncontrolled relations needs to be present in the knowledge graph. This significantly affects the design manipulations. For example, if j and B did not have varying relationships, customer 1 information would remain in the system. Also, the hash values of the entire system would have to be recalculated in case of any changes. Furthermore, since there are no lock and isLock functions in the knowledge graph, the system can only be fixed manually or created from scratch. This can lead to serious time and space losses.

4.2 Reliability

The trust plugin consists of the sum of invariant relations in the knowledge graph. In this way, the trust mechanism allows information to be ranked. Information with a high trust value is more secure and ranks higher.

In the reliability plugin, the reliability of a piece of information is related to the number of immutable relations it has. That is, the more immutable relations a piece of information has with other information, the more reliable it is. It is called suspect information if a piece of information has no fixed relationships. The following example shows a JSON representation of C, which has no fixed relationships. Here the reliability value of C is 0.

C: {{'Image 1'}, Null, Null, {13h}}

The following example shows the JSON format representation of Z information with more than one constant relationship. Here, the reliability value of Z is 2. Regarding reliability, if the user enters a depth parameter, the calculations are made up to that depth. For example, if the depth parameter of Z is 1, the reliability value will also be 1. This feature has been developed to reduce time and space complexity significantly.

k: {{'do'}, Null, Null, Null, {3e4d}}

j: {{'use'}, Null, Null, Null, {2d5S}}

l: {{'sell'}, Null, Null, Null, Null}

M: {{'Customer 3'}, Null, Null, Null, Null}

X: {{'Paint'}, Null, Null, Null, {23ft}}

Y: {{'Image'}, {j: X}, Null, {feSa}}

Z: {{'Painter'}, {k: Y}, {l: M}, {des3}}

There is no practical and simple reliability mechanism in the knowledge graph in the sense we have developed. It is, therefore, not possible to rank trustworthiness. This prevents a trust-based ranking mechanism. On the other hand, the trustworthiness mechanism we have developed can be applied practically and simply to the knowledge graph, thus efficiently addressing the need for trust-based ranking when necessary.

4.3 Consistency

This section explains the consistency extension in the knowledge graph through an example scenario. First, five pieces of information are created. At the time of creation, they have no fixed or variable relationships. Below, the creation of the information is shown in JSON format.

K1: {'A 36-year-old man stabbed his ex-fiancée to death.', Null, Null, Null, Null}

K2: {'23 years in prison sentence requested for man who stabbed his ex-fiancée to death.', Null, Null, Null}

K3: {'Man who stabbed his ex-fiancée to death is released in good condition after first hearing.', Null, Null, Null}

K4: {'Women's rights activists protested this decision in front of the court.', Null, Null, Null}

K5: {'Feminism is spreading.', Null, Null, Null, Null}

Once information is created, a cause-and-effect relationship is established between them. If a piece of information has no relationship, it is not reliable. For example, in the commands below, the cause of the fifth information is the relationship between the fourth, the cause of the fourth is the relationship between the third, the cause of the third is the relationship between the second, and the cause of the second is the relationship between the first. The disappearance of the fourth piece of information would remove the reliability of the fifth piece of information and make it suspect. Below, after the cause and effect relationships of the information have been entered, the relationships between the information are shown in JSON format, locked with the lock function.

A1: {'why', Null, Null, {12fK}}

K1: {'A 36-year-old man stabbed his ex-fiancée to death.', Null, Null, {76Tf}}

K2: {'A man who stabbed his ex-fiancée to death has been sentenced to 23 years in prison.', {A1: K1}, Null {23wS}}

K3: {'The man who stabbed his ex-fiancée to death was released in good condition at the first hearing.', {A1: K2, A1: K1}, Null, {23dS}}

K4: {'Women's rights activists protested this decision in front of the court.',{A1: K3}, Null, {P3se}}

K5: {'Feminism is spreading.', {A1: K4}, Null, {wqq2}}

When an error or change occurs in the information itself or in any of the fixed information that supports it, the model finds the source of the change, removes that source from the context, and updates all the information associated with that source depending on the depth parameter. This ensures consistency in the system.

The consistency concept in the plugin we developed focuses on changes in the copy of the knowledge graph and changes in the knowledge graph itself. A change in any information in the knowledge graph will cause every piece of information in the knowledge graph to be updated and make it possible to update changes in its copies on demand.

4.4 Context

Below, four relationships are created to explain the context of a piece of information.

A1: {{'why'}, Null, Null, Null, Null}

K1: {{'data1'}, Null, Null, Null, Null}

K2: {{'data2'}, A1:K1, Null, Null}

K3: {{'data3'}, {A1:K1, A1:K2}, Null, Null}

K4: {{'data4'}, {A1:K3}, Null, Null}

Below, the context between the above knowledge is expressed. Here, Knowledge1 has no context but persists in the model. This means that there is no invariant relationship to verify Knowledge1. Whether or not any knowledge that has no invariant relationship and is therefore not ordinarily reliable is considered reliable is at the discretion of the creator of the knowledge graph.

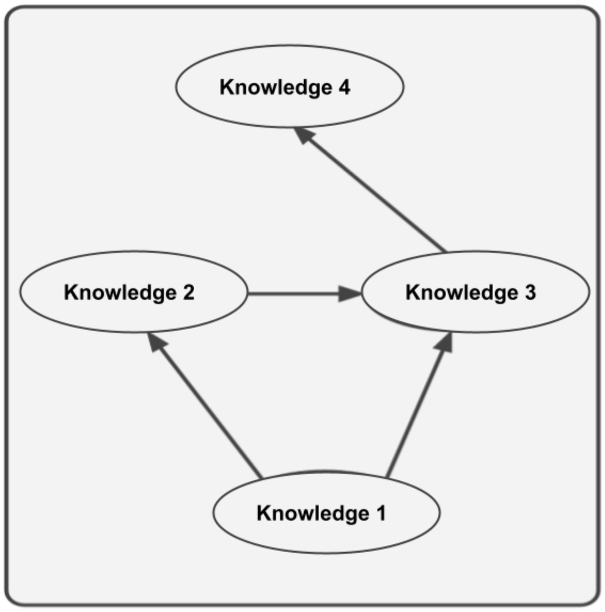

Figure 3. Knowledge's context structure

As seen in Figure 3, the context of Knowledge4 includes Knowledge1, Knowledge2, and Knowledge3. Since the hash value will be specific to the graph when calculating the context, the context of Knowledge4 in the figure above will be specific to Knowledge1, Knowledge2, and Knowledge3 and the relationship between them. As can be seen in Figure 3, the values held in the hash set also determine the context. In other words, since a piece of knowledge can have multiple contexts, it is possible to create the contexts of that knowledge by assigning the desired summaries to the hash set. By looking at these hash values, it can then determine whether one piece of information is compatible with the context of another. This removes many ambiguities about information.

The plugin we developed supports the context mechanism for comparing information in the knowledge graph. This makes it possible to compare information in the knowledge graph easily. As in real life, the value of a piece of information can vary according to many different contexts. This can be easily realized in the plugin we have developed.

5 EXPERIMENT

To test the plugin we developed, we created the experimental setup in Figure 4. In this experiment, persons A, B, C, D, E are created and the relationships between these persons are shown. In the relationships between people, the red arrows cannot be changed and the blue arrows can be changed. For example, being an artist or a father is a fixed relationship, whereas being a moviegoer, the city one lives in, one's friends or hobbies are relationships that can change depending on one's choice. The more fixed relationships person A has, the more trustworthy he/she is considered to be. For example, in Figure 4, A's credibility is 3 and B's credibility is 4. In this case, B is considered more trustworthy than A. Whenever an update is made to B, the constant relations between the labels "female, E and Ankara University" that support B are also updated. This mechanism is not present in the graph data structure.

For example, if we want to change the hobby of cycling, we need to update people D and A who are affected by that hobby. Normally we have to do this manually, which poses a problem for the consistency of the knowledge graph. For example, either we might forget to add the information to the knowledge graph, or we might not add the information in time, which can lead to various problems. This can lead to various inference problems in the knowledge graph. Therefore, an algorithm has been developed that automatically updates any change on the fly. Thus, in the experimental example, A and D were updated automatically.

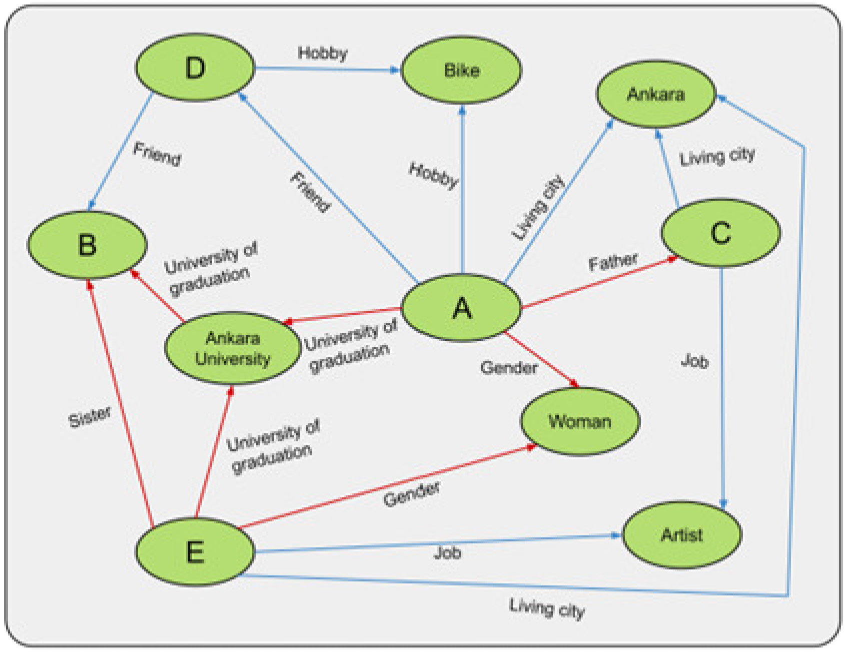

Figure 4. General representation of the experiment

Finally, when we want to compare any two people, for example C and E are two people with the same occupation, living in the same city. If this information is recorded as context, C and E are considered the same person. But C is A's father and E is B's sister. From this point of view, C and E are treated as different people because they have different contexts. Thanks to the hashing mechanism used in our experiments, it is possible to determine in a very short time which contexts they are in and which contexts they have in common.

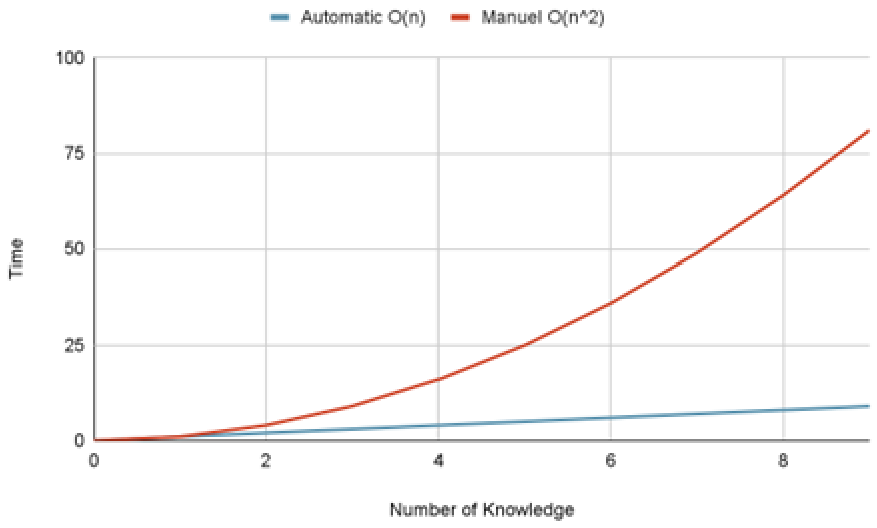

Figure 5. Updating speed of the knowledge graph with improved plugins

Linear time is required to access a piece of information. After accessing the information in question, Deep First Search or Breath First Search must be used to update the values. Since this also takes linear time, a total of O(n^2) time is needed. On the other hand, the algorithm we developed uses only Deep First Search because it is updated instantly, and therefore its complexity is in linear time. The result of the experiments is shown in Figure 5.

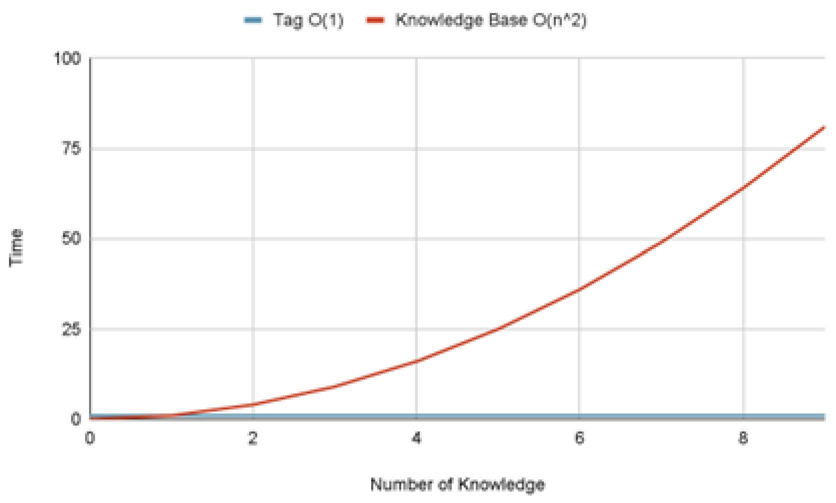

Figure 6. Comparison speed of the knowledge graph with the plugins developed

Finding two pieces of information first and then looking at the properties of the found information leads to an exponential complexity. On the other hand, since our algorithm uses the hashing mechanism, comparing two pieces of information takes place in constant time in Figure 6. This fixed time is the length of the context set.

Table 1. Comparison of Knowledge Graph and Tag mechanism

Type	Update	Immutability	Sorting	Comparison
Knowledge graph	Manual	No	O(n^2)	O(n^2)
Tag	Automatic	There is	O(1)	O(1)

Table 1 shows a comparison with the knowledge graph to illustrate the advantages of the developed plugins. Based on our experiment, it is possible to say that the plugins we have developed provide time benefits by eliminating some important shortcomings in the field of artificial intelligence.

6 EVALUATION

This section presents the advantages and disadvantages of the plugins we developed for the knowledge graph.

The advantage of the immutability extension is that the information in the knowledge graph is stamped as changed/unchanged, making it easy to identify which information has changed. While there is a wide range of work on immutability in the data structure, there is no work on immutability in knowledge graphs.

The immutability plugin contributes to keeping the knowledge graph consistent by allowing information to be easily updated. This contribution is referred to as the consistency plugin in this paper. Thanks to Algorithm 1, the consistency plugin hovers over all changed information and ensures that this information is updated quickly. This function is executed automatically when a piece of information changes and updates all the information it affects based on that change. There is a wide variety of work on consistency in the knowledge graph. However, none of these studies have used a hashing mechanism. At the same time, almost all of the studies in the literature involve very complex procedures.

The credibility plugin allows for the ranking of information. Information ranking reveals the importance of two or more pieces of information. Ranking information in the knowledge graph according to its importance provides the advantage and flexibility to compare the information. Few studies on trustworthiness in knowledge graphs have used hashing mechanisms in the literature. At the same time, almost all existing studies involve rather complex procedures.

The context plugin allows a comparison between two pieces of information. The context plugin allows us to determine whether the information is the same by looking at the hash values. Thanks to the hash set, the information has more than one context, and again thanks to the hash value, the context in which the information is located can be determined. This gives the knowledge graph the advantage of flexibility and abstraction. Moreover, the time complexity is O(1) due to the comparison with the hash algorithm. Although there are several studies on the context in knowledge graphs, none of these studies have used the hashing mechanism. At the same time, almost all existing works involve very complex procedures.

In our work, the disadvantage of the four plugins developed for the knowledge graph is that the hash values of all the information that the knowledge graph is linked to are stored due to the hashing mechanism. Here, a hash value of length N x (256 Bytes) is stored if the information has N links. This slightly increases the space complexity. Another aspect is the runtime of the update function, which is O(E) complexity. The update can be increased or decreased with the diameter parameter. This has a significant impact on the complexity. Considering the contributions of the plugins we have developed to the knowledge graph, it is thought that these two issues, which can be expressed as disadvantages, can be ignored.

7 CONCLUSION

In our work, consistency, context, reliability, and immutability mechanisms are integrated into the knowledge graph in a modular way to perform the operations of where existing knowledge comes from, by whom it is supported, the rate of support, ranking, modifiability or immutability and automatic update. The hashing mechanism was used in the development of these plugins. This is because hashing technology, a straightforward mechanism, offers the possibility to provide four different properties quickly. In our work, a " Knowledge " model provides consistency, immutability, reliability, and context to the knowledge graph.

The first of our proposed extensions, immutability, ensures that when one piece of information is made immutable, all the information associated with it is immutable. This guarantees information reliability. The hash information changes whenever there is a change, so it is immediately possible to identify where the change occurred. The level of trustworthiness is related to the amount of trustworthy information that supports the information. This allows information to be ranked according to its trustworthiness. Consistency refers to the fact that whenever there is a change in the knowledge graph, all affected information is immediately updated. The context consists of all the information related to a piece of knowledge and the relationships between them. The different contexts are calculated and stored in a context array, and the information can be checked for relevance to other contexts by looking at the context array.

With the plugins we have developed, additional features have been added to the knowledge graph, enabling it to reflect knowledge more comprehensively. The extensions are expected to contribute to developing artificial intelligence software that utilizes the knowledge graph. In a broader sense, our work is expected to contribute to developing the software needed in knowledge representation and a wide range of fields related to knowledge, since knowledge is a structure used in every field. In future work, it is planned to realize comprehensive plot implementations of the plugins developed as a proposal.

REFERENCES

Alsaig, A., Alagar, V., and Nematollaah, S. (2020). Contelog: A declarative language for modeling and reasoning with contextual knowledge. *Knowledge-Based Systems*, 207:106403.

Balakrishnan, D., Ziarek, L., and Kennedy, O. (2019). Fluid data structures. In *Proceedings of the 17th ACM SIGPLAN International Symposium on Database Programming Languages*, DBPL 2019, pages 3–17, New York, NY, USA. Association for Computing Machinery.

Bello López, P. and De Ita Luna, G. (2021). An algorithm to belief revision and to verify consistency of a knowledge base. *IEEE Latin America Transactions*, 19(11):1867–1874.

Besançon, L., Da Silva, C. F., Ghodous, P., and Gelas, J.-P. (2022). A blockchain ontology for DApps development. *IEEE Access*, 10:49905–49933.

Cambria, E., Ji, S., Pan, S., and Yu, P. S. (2021). Knowledge graph representation and reasoning. *Neurocomputing*, 461:494–496.

Cano-Benito, J., Cimmino, A., and García-Castro, R. (2021). Toward the ontological modeling of smart contracts: A solidity use case. *IEEE Access*, 9:140156–140172.

Chen, X., Jia, S., and Xiang, Y. (2020a). A review: Knowledge reasoning over knowledge graph. *Expert Syst. Appl.*, 141:112948.

Chen, X., Xie, H., Li, Z., and Cheng, G. (2021). Topic analysis and development in knowledge graph research: A bibliometric review on three decades. *Neurocomputing*, 461:497–515.

Chen, Z., Wang, Y., Zhao, B., Cheng, J., Zhao, X., and Duan, Z. (2020b). Knowledge graph completion: A review. *IEEE Access*, 8:192435–192456.

Chowdhury, M. J. M., Colman, A., Kabir, M. A., Han, J., and Sarda, P. (2018). Blockchain versus database: A critical analysis. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1348–1353. ieeexplore.ieee.org.

Christoforou, E., Nordio, A., Tarable, A., and Leonardi, E. (2021). Ranking a set of objects: A graph based Least-Square approach. *IEEE Transactions on Network Science and Engineering*, 8(1):803–813.

Dai, Y., Wang, S., Xiong, N. N., and Guo, W. (2020). A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5):750.

Delgrande, J. P. and Schaub, T. (2003). A consistency-based approach for belief change. *Artif. Intell.*, 151(1):1–41.

Grant, J., Molinaro, C., and Parisi, F. (2018). Probabilistic spatio-temporal knowledge bases: Capacity constraints, count queries, and consistency checking. *Int. J. Approx. Reason.*, 100:1–28.

Huang, Y., Zhang, L., Yang, X., Chen, Z., Liu, J., Li, J., and Hong, W. (2021). An efficient Graph-Based algorithm for Time-Varying narrowband interference suppression on SAR system. *IEEE Trans. Geosci. Remote Sens.*, 59(10):8418–8432.

Issa, S., Adekunle, O., Hamdi, F., Cherfi, S. S.-S., Dumontier, M., and Zaveri, A. (2021). Knowledge graph completeness: A systematic literature review. *IEEE Access*, 9:31322–31339.

Jabla, R., Khemaja, M., Buendia, F., and Faiz, S. (2022). Automatic rule generation for Decision-Making in Context-Aware systems using machine learning. *Comput. Intell. Neurosci.*, 2022:5202537.

Jiang, S., Liu, Y., Zhang, Y., Luo, P., Cao, K., Xiong, J., Zhao, H., and Wei, J. (2022). Reliable semantic communication system enabled by knowledge graph. *Entropy*, 24(6).

Kejriwal, M. (2022). Knowledge graphs: A practical review of the research landscape. *Information*, 13(4):161.

Khan, N., Ma, Z., Yan, L., and Ullah, A. (2023). Hashing-based semantic relevance attributed knowledge graph embedding enhancement for deep probabilistic recommendation. *Appl Intell (Dordr)*, 53(2):2295–2320.

Liberatore, P. and Schaerf, M. (2001). Belief revision and update: Complexity of model checking. *J. Comput. System Sci.*, 62(1):43–72.

Mu, K. (2015). Responsibility for inconsistency. *Int. J. Approx. Reason.*, 61:43–60.

Mu, K. (2018). Measuring inconsistency with constraints for propositional knowledge bases. *Artif. Intell.*, 259:52–90.

Muiño, D. P. (2011). Measuring and repairing inconsistency in probabilistic knowledge bases. *Int. J. Approx. Reason.*, 52(6):828–840.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2016). A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33.

Nirmala, P. and Nadarajan, R. (2022). Cumulative centrality index: Centrality measures based ranking technique for molecular chemical structural graphs. *J. Mol. Struct.*, 1247:131354.

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., and Taylor, J. (2019). Industry-scale knowledge graphs: Lessons and challenges. *ACM Queue*, 17(2):48–75.

Opdahl, A. L., Al-Moslmi, T., Dang-Nguyen, D.-T., Gallofré Ocaña, M., Tessem, B., and Veres, C. (2022). Semantic knowledge graphs for the news: A review. *ACM Comput. Surv.*, 55(7):1–38.

Ozdayi, M. S., Kantarcioglu, M., and Malin, B. (2020). Leveraging blockchain for immutable logging and querying across multiple sites. *BMC Med. Genomics*, 13(Suppl 7):82.

Porebski, S. (2022). Evaluation of fuzzy membership functions for linguistic rule-based classifier focused on explainability, interpretability and reliability. *Expert Syst. Appl.*, 199:117116.

Rajabi, E. and Etminani, K. (2022). Knowledge-graph-based explainable AI: A systematic review. *J. Inf. Sci. Eng.*, page 01655515221112844.

Ryen, V., Soylu, A., and Roman, D. (2022). Building semantic knowledge graphs from (Semi-)Structured data: A review. *Future Internet*, 14(5):129.

Sciriha, I. and da Fonseca, C. M. (2012). On the rank spread of graphs. *Linear Multilinear Algebra*, 60(1):73–92.

Seo, S., Oh, B., and Lee, K.-H. (2020). Reliable knowledge graph path representation learning. *IEEE Access*, 8:32816–32825.

Stančić, H. and Bralić, V. (2021). Digital archives relying on blockchain: Overcoming the limitations of data immutability. *Computers*, 10(8):91.

Stock, K. and Yousaf, J. (2018). Context-aware automated interpretation of elaborate natural language descriptions of location through learning from empirical data. *Int. J. Geogr. Inf. Sci.*, 32(6):1087–1116.

Terenziani, P. (2000). Integrated temporal reasoning with periodic events. *Comput. Intell.*, 16(2):210–256.

Troussas, C. and Krouska, A. (2022). Path-Based recommender system for learning activities using knowledge graphs. *Information*, 14(1):9.

van Beek, P. and Dechter, R. (1997). Constraint tightness and looseness versus local and global consistency. *J. ACM*, 44(4):549–566.

Verma, S., Bhatia, R., Harit, S., and Batish, S. (2022). Scholarly knowledge graphs through structuring scholarly communication: a review. *Complex Intell Systems*, pages 1–37.

Wang, H., Shang, Y., and Qiao, X. (2020). The integrated organization of data and knowledge based on distributed hash. In *2020 IEEE International Conference on Knowledge Graph (ICKG)*, pages 243–250. ieeexplore.ieee.org.

Wu, W., Zhu, Z., Zhang, G., Kang, S., and Liu, P. (2021). A reasoning enhance network for muti-relation question answering. *Appl. Intell.*, 51(7):4515–4524.

Yang, M., Chen, K., Sun, S., Han, Z., Kong, L., and Meng, Q. (2022). A pattern driven graph ranking approach to attribute extraction for knowledge graph. *IEEE Trans. Ind. Inf.*, 18(2):1250–1259.

Yeh, I., Karp, P. D., Noy, N. F., and Altman, R. B. (2003). Knowledge acquisition, consistency checking and concurrency control for gene ontology (GO). *Bioinformatics*, 19(2):241–248.