Validating ChatGPT Facts by using RDF Knowledge Graphs and Short Sentence Embeddings

Anonymized

No Institute Given

Abstract. Since ChatGPT offers detailed responses without justifications, and erroneous facts even for popular persons, events and places, in this paper we present a novel pipeline that retrieves the response of ChatGPT in RDF and tries to validate the ChatGPT facts using one or more RDF Knowledge Graphs (KGs). To this end we exploit DBpedia and LODsyndesis (a KG comprising more than 400 other RDF KGs), and short sentence embeddings, and provide an algorithm that returns the more relevant triple(s) accompanied by their provenance and a confidence score. To evaluate such services, we create an evaluation benchmark that includes 1,000 ChatGPT facts for famous Greek persons and 500 facts for popular Greek places, which were manually labelled (approximately 75% of ChatGPT facts were correct and 25% of facts were erroneous). The results are promising; indicatively for the collection of persons, we managed to verify the 92.2% of the correct facts of ChatGPT and to find the correct answer for the 57.1% of the incorrect facts of ChatGPT.

Keywords: ChatGPT, LOD, RDF, Knowledge Graphs, Validation

1 Introduction

ChatGPT is a novel Artificial Intelligence (AI) chatbox [26], which is built on GPT-3.5 and GPT-4 families of large language models (LLMs) [6], and provides detailed responses and human-like answers across many domains of knowledge. However, as it is also stated in its webpage "ChatGPT may produce inaccurate information about people, places, or facts" [9]. Concerning issues related to RDF Knowledge Graphs (KGs), ChatGPT can produce erroneous facts and URIs [23], without giving any evidence about the provenance of information. Moreover, it has "limited knowledge of world and events after 2021". On the contrary, it seems that it can successfully produce valid RDF triples by using popular models and ontologies such as the DBpedia ontology, as it is also depicted in Fig. 1. In particular, there are several ways to ask about RDF N-triples and in many cases the triples are exactly the same as in DBpedia (see Fig. 1). On the contrary, there are several issues that can occur and some of them are described below.

First, the triple can be valid but the fact can be erroneous or inaccurate, e.g., see the case of the birth date of the painter "El Greco" or the major of "Naxos"

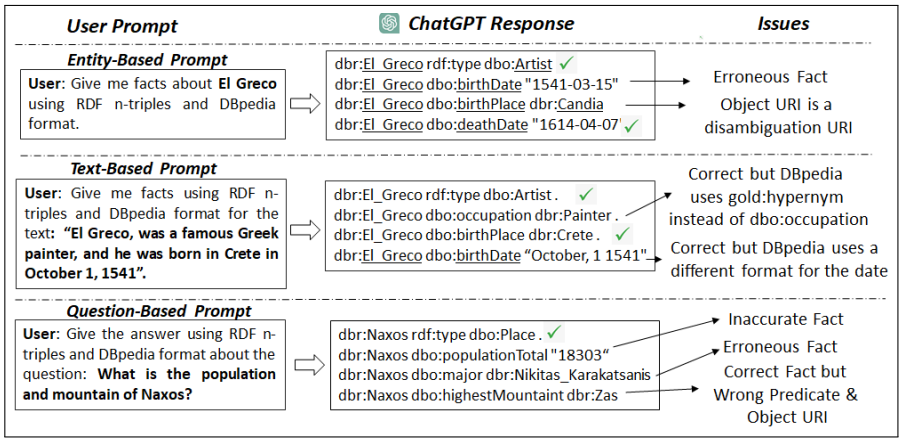

Fig. 1. ChatGPT responses in RDF format using DBpedia Model and Common Errors

island in Fig. 1. Second, the fact can be correct and the URIs valid, however, ChatGPT can fail to produce the correct URIs for the subject, predicate or object, e.g., see that in some cases the Object URI is a disambiguation page. Third, the fact can be correct, but one or more URIs can be invalid or not the appropriate one, e.g., in Fig. 1, the predicates dbo:occupation and dbo:highestMountain are returned from ChatGPT, however for the corresponding triples in DBpedia, the predicates gold:hypernym and dbp:highestMount are used. Finally, in other cases, the literals are not exactly the same, different formats are used for the literals (e.g., see the birth date in the text-based prompt), and literals are used instead of URIs (or the opposite), e.g., for the last fact of Fig. 1, the corresponding triple in DBpedia contains the literal "Mt. Zeus" instead of dbr:Zas.

Concerning the challenges, the objective is how to combine ChatGPT and popular RDF KGs for enabling the validation of ChatGPT facts, given the high quality of information of RDF KGs and that most of them are updated either periodically or continuously [12]. For achieving this target, we need to tackle the mentioned ChatGPT limitations, for finding the correct answer for any ChatGPT fact, i.e., we desire to answer the following research questions (RQs):

- **RQ1**: How to check the validity of ChatGPT facts by using RDF KGs, given the aforementioned difficulties, for validating the correct ChatGPT facts and for finding the correct answer for its erroneous facts?
- **RQ2**: Is it effective to use even more KGs (e.g., through LODsyndesis [20]) for checking the validity of facts instead of using only DBpedia?

As regards our contribution, we propose a novel pipeline that receives as input a ChatGPT response in RDF format and tries to find the most similar fact in a KG for enabling the validation of any ChatGPT fact. For performing this process, we use two different KGs, i.e., DBpedia [18] and LODsyndesis [20] (it contains 400 RDF KGs including DBpedia). The proposed algorithm sends SPARQL queries and REST requests for retrieving candidate similar triples and creates short sentence embeddings for finding the most similar triple to each ChatGPT fact according to the cosine similarity score of vectors. Concerning

the evaluation, we have created a benchmark containing 1,500 RDF triples from ChatGPT, mainly for a list of famous Greek Persons (from the Ancient and Modern era), and popular Greek Places, including Cities, Islands, Lakes, Mountains and Heritage Sites. Moreover, we have manually labelled the ChatGPT facts as correct or erroneous (approximately 25% of ChatGPT facts were erroneous) and we provide an experimental evaluation by comparing the performance of using one or more KGs for fact validation. As regards the results, indicatively, by using LODsyndesis we managed to verify 92.2% of the correct ChatGPT facts for Persons and 77.1% for places, whereas we found the correct answer for the 57.1% the erroneous ChatGPT facts for Persons and 49.1% for places.

Concerning the novelty, to the best of our knowledge, this is the first work trying to validate the facts of ChatGPT by using one or more RDF KGs, whereas it also offers an evaluation benchmark for fact checking over ChatGPT.

The rest of this paper is organized as follows: §2 discusses the background and the related work, §3 introduces the proposed process by showing all the steps and the algorithm for finding relevant facts. Moreover, §4 presents the evaluation benchmark and statistics, whereas §5 presents the experimental evaluation over the benchmark. Finally, §6 concludes the paper and discusses future directions.

2 Background & Related Work

2.1 Background

First, we define a triple as a statement of the form subject-predicate-object ⟨s,p,o⟩, and it is any element of $\mathcal{T}=(\mathcal{U}\cup\mathcal{BN})\times(\mathcal{U})\times(\mathcal{U}\cup\mathcal{BN}\cup\mathcal{L})$, where $\mathcal{U}$, $\mathcal{BN}$ and $\mathcal{L}$ denote the sets of URIs, blank nodes and literals, respectively. Moreover, $T_{KG}\subset\mathcal{T}$ denotes all the triples of a KG, e.g., DBpedia or LODsyndesis.

We denote as $[x]$ the class of equivalence of an element x ($x\in\mathcal{U}$ or $x\in\mathcal{L}$). Specifically, two or more URIs (resources, properties or RDF classes) belong in the same class of equivalence if they refer to the same real world entity (as inferred by the owl:sameAs), property or class, whereas two or more literals are equal, if they use exactly the same string. Moreover, we define that two triples $t=\langle s,p,o\rangle, t'=\langle s',p',o'\rangle$ are equivalent when it holds that $[s]\equiv[s']$, $[p]\equiv[p'], [o]\equiv[o']$. An example of equivalent triples between two KGs, say DBpedia and Wikidata follows: ⟨dbo:Aristotle,dbo:birthDate,"384 BC"⟩ and ⟨wkd:Q868,wkp:P569,"384 BC"⟩, since it holds that ⟨dbp:Aristotle,owl:sameAs, wkd:Q868⟩, ⟨dbo:birthDate, owl:equivalentProperty,wkp:P569⟩ and they use the same literal as object, i.e., "384 BC". Finally, we define all the triples of an entity e in a KG as follows: $T_{KG}(e)=\{\langle s,p,o\rangle\in T_{KG}\mid [s]\equiv e \textit{ or } [o]\equiv e\}$.

DBpedia and LODsyndesis KG. In this paper, we use both DBpedia [18] and LODsyndesis KG [20,22] for enabling the validation of ChatGPT facts. LODsyndesis is an Aggregated Knowledge Graph that contains 2 billion triples from 400 RDF KGs of many domains (including cross-domain KGs like DBpedia, Wikidata [35], YAGO [30], geographical KGs like GeoNames [1], etc.). In particular, LODsyndesis has computed the transitive and symmetric closure of 45

million equivalence relationships, including owl:sameAs, owl:equivalentProperty and owl:equivalentClass relationships, and it has indexed the contents of the 400 RDF KGs, by preserving their provenance. Due to the precomputed closure, it enables the retrieval of equivalent triples to a given triple, which is quite important given that for a fact of ChatGPT, it is possible to find in the KG a semantically equivalent triple and not the exact one.

2.2 Related Work

Here, we focus on approaches that combine ChatGPT and KGs, fact checking using RDF KGs and embeddings over RDF triples. Finally, we provide a comparison and we mention the novelty of the proposed approach.

A. ChatGPT and KGs. First, [24] provides a comparison for the Question Answering task, showing that ChatGPT can have high precision in popular domains but very low precision in unseen domains. Moreover, [23] presents an approach where the response of ChatGPT is annotated by using popular Entity Recognition tools, and each entity is enriched with more statistics and links to LODsyndesis. In [14], the authors evaluated ChatGPT in a named identification task over historical documents, whereas in [15] ChatGPT facts about politics were manually validated. Finally, [25] mentions the importance of providing solutions that combine ChatGPT and RDF KGs, e.g., for improving its accuracy.

B. Fact Checking by using KGs. There is a high trend for automated fact checking approaches by using both textual sources and KGs [36]. Concerning approaches over KGs, [10] used Freebase as a gold standard, for identifying the true values for facts extracted from web pages. Also, DBpedia KG was used in [8] for performing fact checking using simple network models and by using input both natural text (from Wikipedia) and structured data. Furthermore, in [32], DBpedia was also used in an unsupervised network-flow based approach for validating statements, whereas [34] provides a pipeline for explainable fact checking through structured and unstructured data. Moreover, [19] offers a Web service and API that links natural text to a KG of fact-checked claims.

C. Embeddings using RDF Triples (focusing on Fact Checking). There are many approaches that create embeddings from RDF Triples (or/and paths) for several tasks [28], including finding similar entities [31,21] and link prediction [28,4]. Concerning fact checking approaches over RDF, [33] proposed a path-based approach that creates KG embeddings, whereas [29] introduces a hybrid approach which uses both text and KG embeddings by using DBpedia. Also, in [27,3] embeddings from RDF triples were used for fact validation for a Semantic Web Challenge, whereas [16] leverages noisy data from web pages and prior knowledge from KGs (using KG embeddings) for checking the veracity of data. Finally, there are available several evaluation benchmarks for fact checking that are based on popular KGs like DBpedia and Freebase, e.g., [13,17].

Comparison and Novelty. First, compared to approaches over ChatGPT, we mainly focus on enabling the validation of ChatGPT facts, and not on tasks like Entity Recognition [23] and Question Answering [24], or only on performing a manual fact checking evaluation (e.g., [15]). Regarding fact checking, we focus

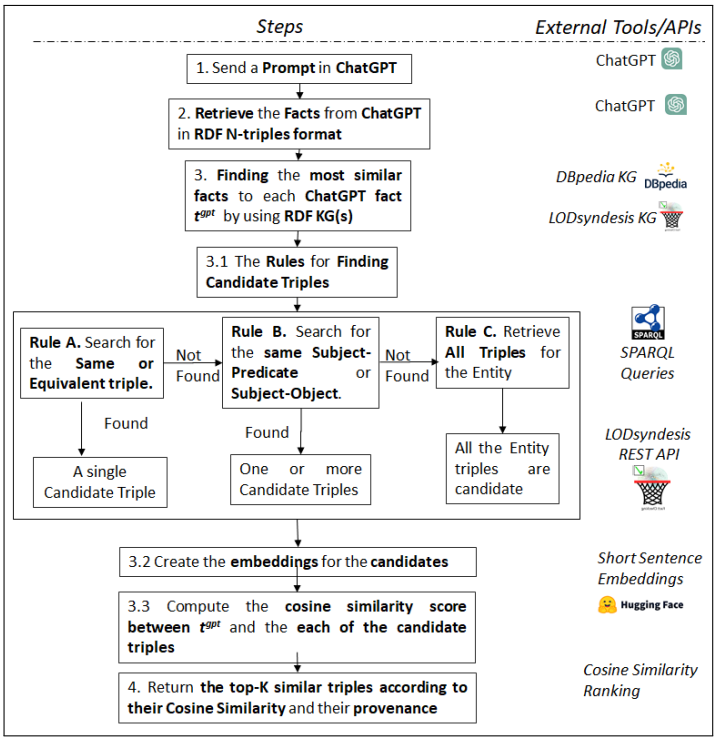

Fig. 2. The Steps of the Proposed Approach and external tools/APIs

on ChatGPT responses and not on data from textual sources (e.g., [10,16]), whereas we provide an approach that combines both SPARQL queries and the creation of embeddings from multiple RDF datasets (through LODsyndesis KG), instead of using a single one like DBpedia (e.g., [8,32]). Finally, our benchmark contains data from ChatGPT and not from other types of sources [13,17].

Regarding the novelty, to the best of our knowledge, this is the first work that a) validates at real time the facts of ChatGPT by using multiple RDF KGs and short sentence embeddings, and by returning the provenance for each fact, and b) offers a benchmark for fact checking over ChatGPT and RDF KGs, and manually annotated results for facts about real persons and places.

3 The Steps of the Proposed Approach

The steps of the process (depicted in Fig. 2) are the following: 1) send a prompt in ChatGPT by asking for RDF triples either from a given response or by asking ChatGPT for facts using the desired format, 2) receive the response as RDF triples, 3) for each fact find the most relevant triple(s) according to three rules by using short sentence embeddings, and rank them in descending order according to their cosine similarity score with the vector of the desired fact and 4) produce the K most similar triples (and their provenance) with their score. The running

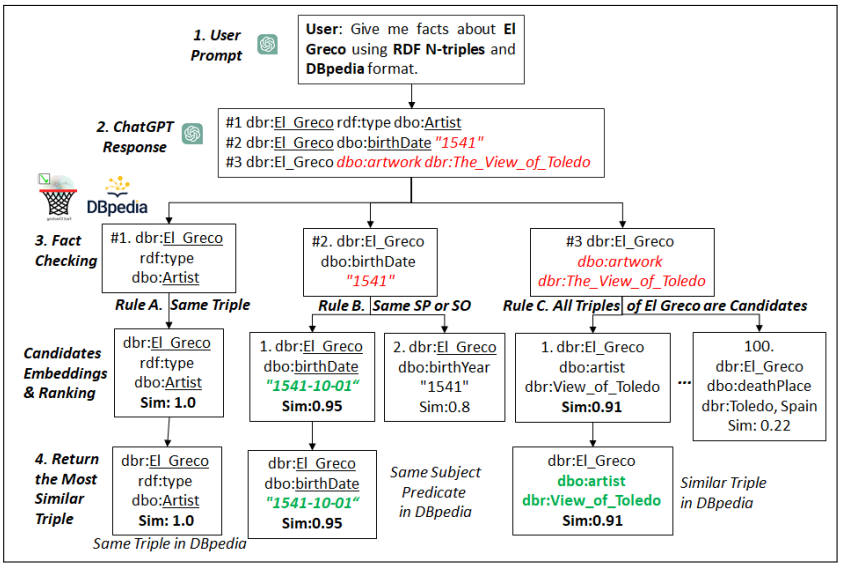

Fig. 3. Running Example by using a real response from the ChatGPT

example of Fig. 3 shows a real response from ChatGPT, the candidate triples for each fact and the K=1 most relevant triple. In particular, the first fact is correct and the triple exists in the KG, the second is inaccurate and the KG contains the same entity-predicate pair and finally the third is correct, however, the correct triple in DBpedia uses different URIs for the predicate and the object.

3.1 Step 1. Send a Prompt to ChatGPT

Here, we provide some key ways to create RDF N-triples by using ChatGPT.

Entity-based Prompt. One can write this prompt: "Give me facts about entity e using RDF N-triples and X format", where e is the name of the entity and X can be the DBpedia format, and it will return the corresponding triples. Such a real case is shown in the upper side of Fig. 1.

Text-based Prompt. One can write this prompt: "Give me facts using RDF N-triples and X format for the text Y", where Y can be any text, such as the response of ChatGPT. An example is shown in the middle side of Fig. 1.

Question-based Prompt. One can write the following prompt: "Give me facts using RDF N-triples and X format about the question q", where q can be any question, e.g., see the lower side of Fig. 1.

3.2 Step 2. Retrieve the Facts from ChatGPT in RDF format

The next step is to collect all the triples from the ChatGPT response, and to analyze each one. In particular, let T_{GPT} be all the triples from a ChatGPT response. For each $t_{gpt} = \langle e, p, o \rangle \in T_{GPT}$, we will check for finding either the same (or an equivalent) triple, a triple with the same subject-predicate or subject-object, or the most similar triple, to a given KG, as it is described in Alg. 1.

Algorithm 1: Finding the most similar fact(s) for Validation

Input: A fact $t^{gpt} = \langle e, p, o \rangle$ from ChatGPT, and a Knowledge Graph KG, and a Sentence Similarity model, denoted by $sentenceModel$

Output: The top-K most similar triples of KG to t^{gpt}

1 $cand(t^{gpt}) \leftarrow \emptyset$

`// Execution of Rules for Finding Candidates`

2 **if** $\exists\ t' \in T_{KG}(e)\ s.t.\ t^{gpt} \equiv t'$ **then** `// Rule A.`

3 $cand(t^{gpt}) \leftarrow \{t'\}$ `// Same or Equivalent Triple found`

4 **else if** $\exists\ t' = \langle e, p', o' \rangle \in T_{KG}(e)\ s.t.\ [p'] = [p]\ or\ [o'] = [o]$ **then** `// Rule B.`

5 $cand(t^{gpt}) \leftarrow \{t' = \langle e, p', o' \rangle \in T_{KG}(e) \mid [p'] = [p]\ or\ [o'] = [o]\}$ `// All the same Subject-Predicate and Subject-Object Pairs are candidates`

6 **else** `// Rule C.`

7 $cand(t^{gpt}) \leftarrow T_{KG}(e)$ `// All the Triples about` e `are Candidates`

`// Create the vector for the ChatGPT fact`

8 $t_c^{gpt} \leftarrow convert(t^{gpt})$ `// Performing Conversions for the URIs`

9 $\vec{t_c^{gpt}} \leftarrow sentenceModel.encode(t_c^{gpt})$ `// Vector for the ChatGPT fact`

`// Creation of vectors and computation of the similarity score`

10 **forall** $t' \in cand(t^{gpt})$ **do** `// for each candidate triple`

11 $t'_c \leftarrow convert(t')$ `// Performing Conversions for the URIs`

12 $\vec{t'_c} \leftarrow sentenceModel.encode(t'_c)$ `// The vector for each candidate by using short sentence embeddings`

13 $\cos(\vec{t_c^{gpt}}, \vec{t'_c}) \leftarrow \frac{\vec{t_c^{gpt}} \cdot \vec{t'_c}}{\|\vec{t_c^{gpt}}\| \|\vec{t'_c}\|}$ `// Computation of Cosine Similarity`

`// Sort the top-`K `similar triples w.r.t. to the cosine similarity`

14 $t_{best} \leftarrow arg_{t'} max\ \{\cos(\vec{t_c^{gpt}}, \vec{t'_c}) \mid t' \in cand(t^{gtp})\}$ `// Most Similar fact(s)`

`// Return the most similar fact(s) and their similarity score`

15 **return** t_{best}, $cos(\vec{t_c^{gpt}}, \vec{t_{best}})$

3.3 Steps 3 & 4. Algorithm for Finding the most similar facts to each ChatGPT facts by using RDF KG(s)

The input for the Alg. 1 is a ChatGPT fact t^{gpt}, the KG that will be used, e.g., DBpedia or LODsyndesis, and the embeddings model that will be used.

Step 3.1. The Rules for Finding Candidate Triples. First, we collect all the candidate triples, i.e., $cand(t^{gpt})$ according to three rules (lines 1-7), which are also shown in Fig. 2 and in the running example of Fig. 3.

Rule A. Search for the same or equivalent triple. We search in the KGs if the same (or an equivalent) triple of t^{gpt} exists, i.e., if $\exists\ t' \in T_{KG}(e)\ s.t.\ t' \equiv t^{gpt}$. In this case, $cand(t^{gpt})$ contains the single triple t' (lines 2-3).

How the Rule is Executed? For checking Rule A, we send an ASK SPARQL query to DBpedia SPARQL endpoint [18] (i.e., when DBpedia KG is used), or a request to the online Fact Checking Service of LODsyndesis REST API[1].

Rule B. Search for the same Subject-Predicate (SP) or Subject-Object (SO). We search for finding triples that have either the same Subject-

[1] `https://demos.isl.ics.forth.gr/lodsyndesis/rest-api`

Predicate (SP) or the Same Subject-Object (SO) Pair, we collect all of them and we use them as candidates, i.e., see lines 4-5 in Alg. 1.

How the Rule is Executed? For collecting the candidates of Rule B, for the case of DBpedia we can send two SPARQL SELECT queries, i.e., one with fixed subject-predicate and one with fixed subject-object. On the contrary, for the case of LODsyndesis, we can send two requests to its Fact Checking service.

Rule C. Retrieve All Triples of e as Candidates. If the previous two rules fail, we use all the triples of the subject e of t^{gpt} (where e occurs either as a subject or as an object), i.e., $cand(t^{gpt}) = T_{KG}(e)$ (see lines 6-7 of Alg. 1).

How the Rule is Executed? For collecting the candidates, we either send a SELECT SPARQL query for retrieving all the facts for the entity e, or a REST request to the "allFacts" service of LODsyndesis REST API.

Step 3.2 Creating the Embeddings. For the next steps (lines 8-13 of Alg. 1), we create a vector for both the t^{gpt} and each $t^{gpt} \in cand(t^{gpt})$.

Short Sentence Embeddings. Since each triple is a short statement, and given the detected problems from ChatGPT (see Fig. 1), e.g., different predicates, literals instead of URIs, wrong and invalid URIs, including properties, etc., we decided to use sentence similarity models. Indeed, "Sentence Similarity is the task of determining how similar two texts are, by converting input texts into vectors (embeddings) that capture semantic information and calculate how close (similar) they are between them". Here, we use the most popular Sentence Similarity model from huggingface, called "sentence-transformers/all-MiniLM-L6-v2"[11], which "maps sentences and paragraphs to a 384 dimensional dense". The objective is to measure how similar each candidate triple is with the fact t^{gpt} in terms of their semantic meaning. However, such models need to process natural text and not triples, thereby, some URI conversions are needed.

Conversions for URIs. For the URIs of t^{gpt} and of each $t' \in cand(t^{gpt})$, we perform some conversions, i.e., we remove the prefixes and the special characters from the URIs such as the underscore "_". Moreover, in several cases, capital letters are used for distinguishing two or more words, e.g., dbo:associatedBand, and in such cases we split the words when a capital letter is identified, e.g., the previous will be converted to "associated Band". On the contrary, for the URIs whose suffix is not human-readable (e.g., Wikidata), we replace the URIs with their labels. As an example, the triple "dbr:El_Greco, dbo:artist dbr:View_of_Toledo", will be converted to "El Greco artist View of Toledo", however, we also keep the initial triple and its provenance (i.e., for returning it as the final output).

Creation of Vectors for the Converted Triples. First we create the vector for the converted ChatGPT fact t_c^{gpt}, i.e., we create $\vec{t_c^{gpt}}$. Afterwards, for each $t' \in cand(t^{gpt})$, we create a vector for its converted version, i.e., $\vec{t'_c}$.

Step 3.3 Computation of their Cosine Similarity with triple t^{gpt}. For each t' we compute the cosine similarity with the triple t^{gpt} (line 13). We selected to use cosine similarity, since it is has been successfully (and widely) used for RDF similarity-based approaches based on embeddings (e.g., see [7,31]). The

Part	Description of Entities that we asked ChatGPT	Facts	Correct Facts (percentage)	Erroneous Facts (percentage)	Unique URIs (dereferencable)	Unique Properties (dereferencable)
Greek Persons	From the Great Greeks List	1000	812 (81.2%)	188 (18.8%)	525 (472)	109 (85)
Greek Places	Heritage Sites, Cities, Islands, Lakes, Mountains	500	319 (63.8%)	181 (36.2%)	219 (177)	85 (56)
Total	-	1500	1131 (75.4%)	369 (24.6%)	744 (649)	194 (141)

Table 1. The Evaluation Benchmark and Statistics

cosine similarity ranges from -1 (i.e., the two vectors are exactly opposite) to 1 (i.e., the two vectors are exactly the same, thereby the triples are equivalent).

Step 4. Return the top K similar triples (and provenance). The last step is to sort the candidate triples according to their cosine similarity, for returning the top-k results (i.e., see lines 14-15). Apart from the best triple(s), we return the cosine similarity score and the provenance of each triple. Concerning the value of K, some suggested values are $1, 3, 5$. Certainly, in some cases we have only one candidate, e.g., always in cases when Rule A is followed (e.g., see the example of Fig. 3), and in many cases either one or a few candidates, i.e., when Rule B is executed (e.g., see the example of Fig. 3). On the contrary, for the rule C, we always have a larger number of candidates, since we compute the similarity between t^{gpt} and all the triples of the entity, e.g., see the example of Fig. 3. In that case, we successfully verified the ChatGPT fact, although the KG used a different predicate and object URI for expressing the same real fact.

Time and Space Complexity of Alg. 1. In the worst case, we need to iterate over all the triples $T_{KG}(e)$ for the subject e of the fact t^{gpt}, i.e., for creating the vector of each candidate and to compute its cosine similarity. Then we need to sort them according to their similarity score thereby the time complexity is $O(n * log(n))$, where n refers to $T_{KG}(e)$ in our case. On the contrary, we load in memory the sentence similarity model, and we keep in memory all the candidate triples and their cosine similarity score, thereby the space complexity is $O(n+m)$ (n refers to $T_{KG}(e)$, and m to the size of the sentence similarity model).

4 Benchmark for ChatGPT Facts & RDF KGs

We collected a list of famous Greek Persons and Places (i.e., since the current edition of the conference is in Greece). For the persons, we used the list of the 100 Greatest Greeks of all time [2], including persons from different eras (Ancient and Modern era) and disciplines (leaders, artists, etc.). For the places, we used a list of famous heritage sites and islands, and the largest cities, lakes and mountains.

Collections of Facts from ChatGPT and Manual Annotation. For each entity, we send the following requests to ChatGPT (using "gpt-3.5-turbo", March 23 version) as follows: "Give me facts in RDF N-Triples format for entity e

ID	Predicate of Fact from ChatGPT	Total	Correct Facts	Erron. Facts
1	dbo:occupation	102	97	5
2	dbo:birthDate	90	60	30
3	rdf:type	85	85	0
4	dbo:deathDate	83	63	20
5	dbo:birthPlace	78	61	17
6	dbo:nationality	74	74	0
7	dbo:deathPlace	69	63	6
8	dbo:influenced	33	19	14
9	dbo:notableWork	21	17	4
10	dbo:field	20	17	3

Table 2. Top-10 Most Used Predicates for Greek Persons

ID	Predicate of Fact from ChatGPT	Total	Correct Facts	Erron. Facts
1	rdf:type	56	56	0
2	dbo:country	51	51	0
3	dbo:elevation	36	11	25
4	dbo:location	21	20	1
5	dbo:population	19	4	15
6	dbo:areaTotal	19	0	19
7	dbo:timeZone	17	17	0
8	dbo:settlement	17	14	3
9	dbo:locatedIn	15	15	0
10	dbo:length	14	1	13

Table 3. Top-10 Most Used Predicates for Greek for Places

using DBpedia format", where e is replaced by the name of the person (like Aristotle) or of the place (like Santorini). The resulting benchmark contains 1,500 facts as it can be seen in Table 1, which also shows some statistics about the collection. The collection can be accessed through `https://anonymous.4open.science/r/ChatGPT-Fact-Validation-using-RDF-Knowledge-Graphs-07C3`.

Then, we collected and manually labelled the facts of ChatGPT as correct or erroneous (annotation conducted in April 2023). We annotated as a) *Correct Facts*, the ChatGPT facts that can be verified from online sources by searching on the web (e.g., Wikipedia, DBpedia, etc.), and b) as *Erroneous Facts* (including inaccurate ones, e.g., small differences in numbers), the ChatGPT facts where we found the correct answer on the web (and it was a different one) or/and facts that cannot be verified from online sources, i.e., we did not find any evidence.

Statistics. Table 1 introduces the collection by showing the number of facts, how many of them were correct or erroneous, the number of unique URIs and properties and whether the produced triples contain dereferencable URIs. As we can see, the facts for the people were more accurate compared to the places, i.e., for the people approximately 1 out of 5 facts was erroneous, whereas for the places, almost 2 out of 5. Concerning URIs describing resources, 90% for Persons and 80% for Places are dereferencable, whereas for Properties, the corresponding percentages are 77% for Persons and 66% of Places. For measuring whether a URI is dereferencable, we send an ASK query for checking if exists at least one triple including the desired URI (resource or property).

Most Frequent Predicates and Erroneous Facts. Concerning the most frequent predicates, for the persons, i.e., see Table 2, they mainly describe general information such as the occupation of persons, their birth/death date and birth/death place and others. Concerning the erroneous facts, they were mainly facts related to numbers and dates, such as "birth/death dates", but also some other ones like "birth and death place" of a person. Concerning other frequent erroneous cases, they include the predicates "influenced", "education" and "child". On the contrary, several correct facts were indicatively about the "type", "nationality", 'notable works" and "fields" of a person. Regarding the places (see

ID	ChatGPT fact	Top Fact (LODsyndesis)	Annotation	Rule Used
1	dbr:Aristophanes dbo:genre dbr:Comedy	dbr:Aristophanes wkd:P136 dbr:Comedy	C1. Correct & Validated	A (equivalent triple)
2	dbr:Georgios_Papanikolaou dbo:knownFor "Pap test"	dbr:Georgios_Papanikolaou dbo:knownFor dbr:Pap_Smear	C1. Correct & Validated	B (same predicate)
3	dbo:Aristophanes dbo:notableWorks dbr:Lysistrata	dbo:Aristophanes dbo:author dbr:Lysistrata	C1. Correct & Validated	B (same object)
4	dbr:Pericles dbo:office dbr:Strategos	dbr:Pericles yago:rank "Strategos"	C1. Correct & Validated	C (most similar triple)
5	dbr:Papadiamantis dbo:notableWork dbr:The_Murderess	dbr:Papadiamantis rdf:type dbo:Writer	C2. Correct w/o Validation	C (most similar triples)
6	dbr:Georgios_Papanikolaou dbo:birthDate "1886-05-13"	dbr:Georgios_Papanikolaou dbo:birthDate "1883-05-13"	C3. Erroneous & Validated	B (same predicate)
7	dbr:Mikis_Theodorakis dbo:activeYears "1949"	dbr:Mikis_Theodorakis dbo:yearsActive "1943"	C3. Erroneous & Validated	C (most similar triples)
8	dbr:Charilaos_Florakis dbo:child dbr:Artemis_Floraki	dbr:Charilaos_Florakis yago:familyName "florakis"	C4. Erroneous w/o Validation	C (most similar triples)

Table 4. Real Examples of Labels (Annotations) from the Evaluation Benchmark

Table 3), the most erroneous ChatGPT facts contain numbers, e.g., "elevation", "population", "areaTotal", "length", "width", "depth", whereas it also failed in many cases to find the "major" of specific cities. On the contrary, ChatGPT responses were more accurate for properties like the "type", "location", "country" and "timeZone". Finally, concerning other inaccurate cases, they include "owl:sameAs" relationships to RDF KGs that use identifiers for their URIs (e.g., Wikidata), and properties with identifiers as values, e.g., "wikiPageRevisionID".

5 Experimental Evaluation

Here, we use the evaluation benchmark of §4, for evaluating mainly the effectiveness of the proposed approach, however, we also provide the execution times. Our target is also to compare the performance of using only a single KG, i.e., DBpedia versus more KGs, i.e., LODsyndesis (where DBpedia is also included). For performing the experiments we used a *Backend Python 3 Google Compute Engine from Google Colab* with 12.7 GB RAM and 107.7 GB disk space.

5.1 The Process, Manual Labelling & Metrics

For each fact of the benchmark, we find the $K = 3$ most similar triples. We decided to use $K = 3$, since in many cases the top-1 result can be correct but not the desired one, especially for multi-valued predicates. For instance, in an erroneous fact saying that the birth place of a person is the city of "Thessaloniki", the algorithm returned as the top similar fact from DBpedia that his birth place is "Greece" and as a second similar that is the city of "Larissa". Although both triples are correct, the second is the desired one for the mentioned case. However, remind that when Rule A is used, there is only one result, whereas when rule B is used, the algorithm (when $K = 3$) can return either 1, 2 or 3 results. For providing accurate results, we manually annotated each pair of a ChatGPT triple and the returned answer of the used KG. The 4 categories are listed below:

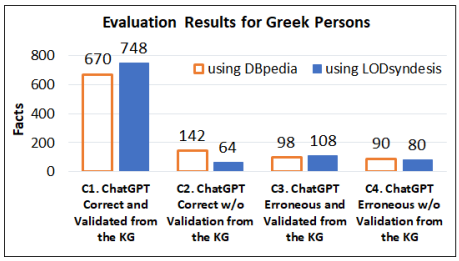

Fig. 4. Evaluation Results for Persons

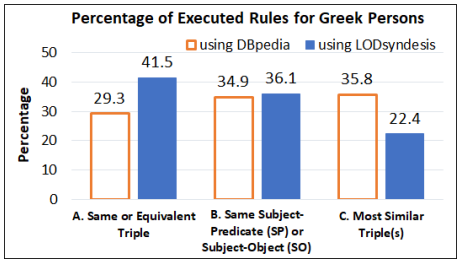

Fig. 5. Rules Executed for Persons

• C1.*Correct & Validated*: Correct in ChatGPT and validated from the KG, even by using a different predicate and object, i.e., see the IDs 1-4 in Table 4. For example, for ID 1, the triples are equivalent since both "dbo:genre" and "wkd:P136" refer to the characteristic "genre".

• C2.*Correct w/o Validation*: Correct in ChatGPT, but not validated from the KG (maybe the KG did not contain the fact or Alg. 1 failed to find it), e.g., in ID 5 of Table 4 the fact is correct, however, the KG does not include it.

• C3. *Erroneous & Validated*: Erroneous in ChatGPT, but the KG provided the correct answer for the same real fact, e.g., see the IDs 6-7 in Table 4.

• C4.*Erroneous w/o Validation*: Erroneous in ChatGPT and the KG failed to provide the correct answer (maybe the KG did not contain the fact or included an erroneous value for the fact, and cases where the algorithm failed to find it), e.g., suppose that for some persons can produce facts that they had children (although they did not). For instance, see the ID 8 in Table 4, where the fact is incorrect, however, LODsyndesis does not include a relevant fact, since the greek politician "Charilaos Florakis" did not have children.

Metrics. We count the number of facts of each category, by using i) only DBpedia and ii) LODsyndesis. The goal is to evaluate the *RQ1*, i.e., we expect to observe high numbers for C1 and C3 (i.e., the KG(s) provided the correct answer) and low numbers for C2 and C4 (i.e., the KG(s) failed to find the correct answer), and the *RQ2*, i.e., we expect better results by using multiple RDF KGs.

Note. We should note that for (some very few) facts, where in the top-3 triples we found contradicting values from many KGs (e.g., for the population of a place), we labelled them as not validated (categories C2 and C4).

5.2 Validation of Facts for Persons

As we can see in Fig. 4 we managed to validate 748 correct facts (out of 812) by using LODsyndesis, i.e., 92.2% of the correct ChatGPT facts, whereas by using only DBpedia, we validated 670 correct facts, i.e., 82.6%. As regards the erroneous ChatGPT facts, we validated (i.e, we found the correct answer) 108 (out of 188) facts by using LODsyndesis, i.e., 57.1%, and 98 erroneous facts by using only DBpedia, i.e., 51.8%. By using LODsyndesis we managed to validate more facts, since some facts that are not described in DBpedia, were found in other

Rule	C1. Correct and Validated	C2. Correct w/o Validation	C3. Erron. and Validated	C4. Erron. w/o Validation	Total
A. Same/Equivalent Triple	415	0	0	0	415
B. Same Subject Predicate	108	23	84	23	238
B. Same Subject Object	117	3	2	1	123
C. Most Similar Triples	108	38	22	56	224
Total	748	64	108	80	1000

Table 5. Analysis of Results (based on Rules) for Greek Persons Using LODsyndesis

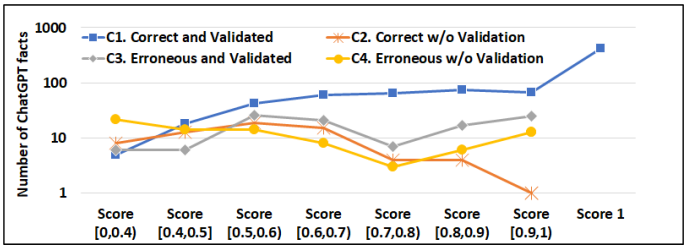

Fig. 6. Cosine Similarity and Annotations for the Persons using LODsyndesis

popular KGs such as Wikidata [35], YAGO [30] and Freebase [5]. This can be explained through Fig. 5, which shows the percentage of rules that were executed for each KG. In particular, by using LODsyndesis in 41.5% of cases we found the same or equivalent triple (due to the precomputed closure), whereas the corresponding percentage for DBpedia was 29.3%. Moreover, by using LODsyndesis in 36.1% of cases we found at least the same SP or SO Pair and only in 22.4% of cases the rule C executed. On the contrary, for DBpedia, in 34.9% of cases we found the same SP or SO Pair, and in 35.8% of cases the rule C executed.

Analysis of Rules and Labelled Results by using LODsyndesis. In Table 5 we analyze the results by using LODsyndesis for the three rules. When rule A was executed all the facts were correct, whereas for rule B, in almost all cases, when we found the same SO, the ChatGPT facts were correct. On the contrary, when we found the same SP, in many cases the fact was erroneous. Indeed, from the 108 erroneous facts that we found the correct answer, 84 of them were verified by using the mentioned rule. Finally, concerning Rule C, in several cases we found the correct answer (categories C1 and C3), however, it was also the most frequent rule when we failed to find the correct answer.

Cosine Similarity and Annotations. Fig. 6 shows that for low cosine similarity scores (between the ChatGPT fact and the KG) less facts were validated, whereas for high scores, more facts were verified. It is worth noting that in some cases, a fact in the KG was correct and in ChatGPT erroneous, however, their similarity was very high, e.g., for similar numbers, dates and places.

5.3 Validation of Facts for Places

For this collection, the results are shown in Fig. 7. Concerning the correct facts we managed to verify 238 (out of 319) by using DBpedia, i.e., 74.6%, and 246 of

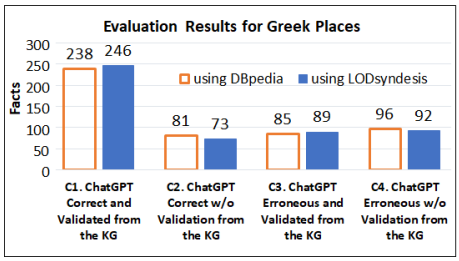

Fig. 7. Evaluation Results for Places

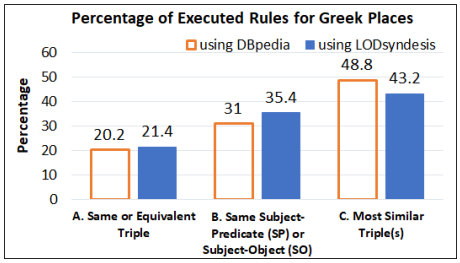

Fig. 8. Rules Executed for Places

them by using LODsyndesis, i.e., 77.1%. For the erroneous facts, we verified 85 (out of 181) through DBpedia (i.e., 46.1%) and 89 of them using LODsyndesis (i.e.,49.1%) In that case, the differences are small, since the other KGs included in LODsyndesis did not provide so many complementary data for the entities of this collection (compared to the persons collection). Concerning the executed rules (see Fig 8) for this collection we found less equivalent triples and SP or SO Pairs, thereby, Rule C was executed more times compared to the persons.

5.4 Discussion and Limitations

Concerning the *RQs*, we verified most of the correct ChatGPT facts by using the proposed pipeline (i.e., *RQ1*), whereas by using multiple KGs we observed better results (i.e., *RQ2*). As regards erroneous ChatGPT facts that Alg. 1 found the correct answer, they mainly include facts about the birth/death date or place of a person, education of persons, facts concerning the predicate influenced, the elevation of places, the major of cities and others. For cases where we failed to identify the correct answer, the most common follow: a) the answer was included in large literals (e.g., abstracts), and the similarity score with such triples and the desired fact was low, b) predicates with very similar suffix, e.g., "population","populationTotal", "populationAsOf", c) contradicting values among KGs (by using LODsyndesis), and d) incompleteness issues, i.e., cases where the KG(s) did not contain the fact, e.g., awards, missing birth date and place, etc.

For overcoming such limitations, a possible research direction could be to provide an extended version of Alg. 1, i) for supporting text extraction from large literals (e.g., through Question Answering techniques), ii) for resolving the conflicts between different KGs (e.g., for predicates and objects) and iii) for performing web search using a search engine (for tackling incompleteness issues).

5.5 Efficiency Results

Fig. 9 shows the average execution time per fact by using a) DBpedia and b) LODsyndesis, whereas Fig. 10 shows the average execution time per rule. Specifically, for the persons the average execution time was lower, i.e., since the rule C, which is the most time consuming one (because it includes the creation of the

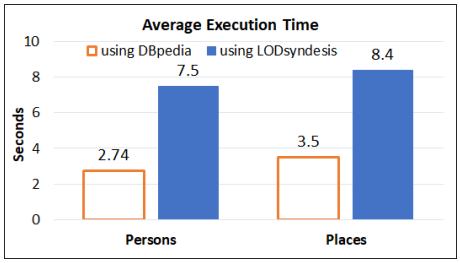

Fig. 9. Average Execution Time for each KG and each Collection

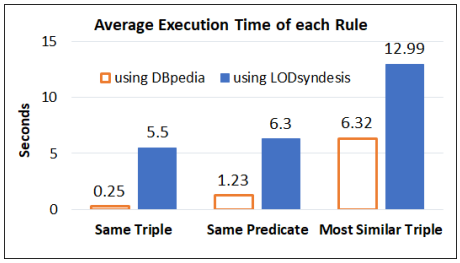

Fig. 10. Average Execution Time for each Rule and KG

embeddings), was executed more times for the places collection. On the contrary, for both collections, it was much faster to use only DBpedia, since it contains a smaller number of triples for each entity, compared to LODsyndesis. Concerning the total execution time, by using DBpedia we needed 45 min for all the facts for the persons and 29 min for the places, whereas for LODsyndesis the corresponding times were 125 min for the persons and 70 min for the places.

6 Concluding Remarks

Since ChatGPT offers detailed responses without evidence, and erroneous facts even for popular persons, events and places, in this paper we presented a novel pipeline that retrieves the response of ChatGPT in RDF format and tries to validate the ChatGPT facts using one or more RDF Knowledge Graphs. For performing this task, we exploited DBpedia and LODsyndesis KG (which includes 400 RDF KGs), and short sentence embeddings. Afterwards, we created an evaluation benchmark including 1,000 ChatGPT facts for famous Greek persons and 500 facts for popular Greek places, which were manually labelled (approximately 75% of ChatGPT facts were correct and 25% of facts were erroneous).

Concerning the results, by using LODsyndesis we verified 92.2% of the correct ChatGPT facts (+9.6% more compared to using only DBpedia) for the persons and 77.1% of the correct ChatGPT facts for places ChatGPT (+2.5% more compared to DBpedia). For the erroneous facts, we found the correct answer in 57.1% of facts for persons and 49.1% for places through LODsyndesis (+5.3% for persons and +2.2% for places versus DBpedia). As a future work, we plan to i) create a web application and a REST API by using the proposed pipeline, ii) extend the benchmark with more entities, facts and domains, iii) propose methods for automating the annotation/evaluation of ChatGPT facts, iv) extend the algorithm for supporting answer extraction from large literals and web search, and v) use/evaluate more sentence similarity models.

Supplemental Material Statement: The source code (python), the evaluation benchmark, including its initial and its human labelled version, experimental results and statistics are available (anonymized) in `https://anonymous.4open.science/r/ChatGPT-Fact-Validation-using-RDF-Knowledge-Graphs-07C3`.

References

1. GeoNames geographical database. http://www.geonames.org/, accessed: May, 4, 2023
2. Great Greeks. https://en.wikipedia.org/wiki/Great_Greeks, accessed: May, 4, 2023
3. Ammar, A., Celebi, R.: Fact validation with knowledge graph embeddings. In: ISWC (Satellites). pp. 125–128 (2019)
4. Biswas, R.: Embedding based link prediction for knowledge graph completion. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 3221–3224 (2020)
5. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250 (2008)
6. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
7. Chatzakis, M., Mountantonakis, M., Tzitzikas, Y.: Rdfsim: similarity-based browsing over dbpedia using embeddings. Information **12**(11), 440 (2021)
8. Ciampaglia, G.L., Shiralkar, P., Rocha, L.M., Bollen, J., Menczer, F., Flammini, A.: Computational fact checking from knowledge networks. PloS one **10**(6), e0128193 (2015)
9. van Dis, E.A., Bollen, J., Zuidema, W., van Rooij, R., Bockting, C.L.: Chatgpt: five priorities for research. Nature **614**(7947), 224–226 (2023)
10. Dong, X.L., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., Zhang, W.: Knowledge-based trust: Estimating the trustworthiness of web sources. arXiv preprint arXiv:1502.03519 (2015)
11. Face, H.: sentence-transformers/all-minilm-l6-v2 (2023), https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2/ [Accessed: (May, 4, 2023)]
12. Färber, M., Bartscherer, F., Menne, C., Rettinger, A.: Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. Semantic Web **9**(1), 77–129 (2018)
13. Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngomo, A.C.N., Speck, R.: Defacto—temporal and multilingual deep fact validation. Journal of Web Semantics **35**, 85–101 (2015)
14. González-Gallardo, C.E., Boros, E., Girdhar, N., Hamdi, A., Moreno, J.G., Doucet, A.: Yes but.. can chatgpt identify entities in historical documents? arXiv preprint arXiv:2303.17322 (2023)
15. Hoes, E., Altay, S., Bermeo, J.: Using chatgpt to fight misinformation: Chatgpt nails 72% of 12,000 verified claims (2023)
16. Huang, J., Zhao, Y., Hu, W., Ning, Z., Chen, Q., Qiu, X., Huo, C., Ren, W.: Trustworthy knowledge graph completion based on multi-sourced noisy data. In: Proceedings of the ACM Web Conference 2022. pp. 956–965 (2022)
17. Huynh, V.P., Papotti, P.: Towards a benchmark for fact checking with knowledge bases. In: Companion Proceedings of the The Web Conference 2018. pp. 1595–1598 (2018)
18. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic web **6**(2), 167–195 (2015)

19. Maliaroudakis, E., Boland, K., Dietze, S., Todorov, K., Tzitzikas, Y., Fafalios, P.: Claimlinker: Linking text to a knowledge graph of fact-checked claims. In: Companion Proceedings of the Web Conference 2021. pp. 669–672 (2021)
20. Mountantonakis, M.: Services for Connecting and Integrating Big Numbers of Linked Datasets, vol. 50. IOS Press (2021)
21. Mountantonakis, M., Tzitzikas, Y.: Knowledge graph embeddings over hundreds of linked datasets. In: Metadata and Semantic Research: 13th International Conference, MTSR 2019, Rome, Italy, October 28–31, 2019, Revised Selected Papers. pp. 150–162. Springer (2019)
22. Mountantonakis, M., Tzitzikas, Y.: Content-based union and complement metrics for dataset search over rdf knowledge graphs. ACM JDIQ **12**(2), 1–31 (2020)
23. Mountantonakis, M., Tzitzikas, Y.: Using multiple RDF knowledge graphs for enriching ChatGPT responses. In: Demo paper, submitted (2023)
24. Omar, R., Mangukiya, O., Kalnis, P., Mansour, E.: Chatgpt versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots. arXiv preprint arXiv:2302.06466 (2023)
25. Ontotext: Why should you combine chatgpt with knowledge graphs? (2023), `https://www.ontotext.com/blog/why-should-you-combine-chatgpt-with-knowledge-graphs/` [accessed: (May, 4, 2023)]
26. OpenAI: Chatgpt. `https://openai.com/blog/dall-e-3-chat/` (2021), accessed: March 21, 2023
27. Pister, A., Atemezing, G.A.: Knowledge graph embedding for triples fact validation. In: ISWC (Satellites). pp. 21–24 (2019)
28. Portisch, J., Heist, N., Paulheim, H.: Knowledge graph embedding for data mining vs. knowledge graph embedding for link prediction–two sides of the same coin? Semantic Web **13**(3), 399–422 (2022)
29. Qudus, U., Röder, M., Saleem, M., Ngonga Ngomo, A.C.: Hybridfc: A hybrid fact-checking approach for knowledge graphs. In: The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings. pp. 462–480. Springer (2022)
30. Rebele, T., Suchanek, F., Hoffart, J., Biega, J., Kuzey, E., Weikum, G.: Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In: The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II 15. pp. 177–185. Springer (2016)
31. Ristoski, P., Rosati, J., Di Noia, T., De Leone, R., Paulheim, H.: Rdf2vec: Rdf graph embeddings and their applications. Semantic Web **10**(4), 721–752 (2019)
32. Shiralkar, P., Flammini, A., Menczer, F., Ciampaglia, G.L.: Finding streams in knowledge graphs to support fact checking. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 859–864. IEEE (2017)
33. da Silva, A.A.M., Röder, M., Ngomo, A.C.N.: Using compositional embeddings for fact checking. In: The Semantic Web–ISWC 2021: 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24–28, 2021, Proceedings 20. pp. 270–286. Springer (2021)
34. Vedula, N., Parthasarathy, S.: Face-keg: Fact checking explained using knowledge graphs. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining. pp. 526–534 (2021)
35. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)
36. Zeng, X., Abumansour, A.S., Zubiaga, A.: Automated fact-checking: A survey. Language and Linguistics Compass **15**(10), e12438 (2021)