RDF Graph GPT - Building knowledge graphs with Chat GPT

Juana Borrelli Zara [1] [0009-0004-3686-182X] ,Alejandro Fernández [1] [0000-0002-7968-6871] and Diego Torres [1, 2] [0000-0001-7533-0133]

[1] LIFIA,CICPBA-Facultad de Informática, Universidad Nacional de La Plata 50 y 120, La Plata, Buenos Aires, Argentina.
jbzara@lifia.info.unlp.edu.ar
casco@lifia.info.unlp.edu.ar
[2] Depto CyT, Universidad Nacional de Quilmes Roque Saenz Peña 352, Bernal, Buenos Aires, Argentina.
dtorres@lifia.info.unlp.edu.ar

Abstract. This paper introduces RDF Graph GPT app as a tool to automatically translate natural language texts into RDF format knowledge graphs using Artificial Intelligence. Throughout its reading we will dissect and elaborate the various aspects of the functionality mentioned above, as well as some more that this app provides such as graph visualization of any RDF format text with the possibility to choose between two graph types. It also presents an overview of the connection with the AI, more precisely Chat GPT, and the prompts structure that leads to the desired results. The app was tested using different types of natural language texts and different prompts in order to make a richer analysis of the results. This evaluation process was made by experts who gave us some conclusions about the scope and limitations of the tool. It has promising results that show us how interesting this task is along with the potential that the resultant app has. This app, then, puts the focus on the IA as a resource on the matter of developing powerful tools for translating natural language texts into structured formats such as RDF. A very desired task to do automatically due to its implications and applications in the matter of adding semantics to the data.

Keywords: Resource Description Framework (RDF), Large Language Models (LLMs), Artificial Intelligence, Chat GPT, Knowledge Graph.

1 Introduction

The Semantic Web is a visualization of the Web in which the data is structured and presented in a certain way to make it more machine-readable [1]. This is of great significance when it comes to automatic tasks and intelligent web searches. Machines are good in automatic and repetitive tasks where humans tend to make mistakes; humans, on the other hand, have the ability to understand the semantics of language, as long as computers only see it as symbol concatenation. As a result, the realization of the

Semantic Web vision is led by a collection of standards and techniques to make data structured [2].

One of the most powerful techniques to make data machine-readable is the natural language text translation into knowledge graphs, whose nodes represent the text entities and whose edges represent the relationships between them [3].There are various graph models, such as Directed Edge-labelled Graphs in which we find nodes and binary relations between them, where we can recognize the subject, from which the arrow will start, and the object, where the arrow will aim, and finally, the predicate will be the arrow's label. A standard data model based on this kind of graph is the Resource Description Framework (RDF) used for globally identifying entities and relations on the Web [4].

That is, then, the motivation of this article, given an unstructured text in natural language, to obtain a correct knowledge graph in RDF format. We can visualize this motivation in the translation from Listing 1 to Listing 2. There are important characteristics of natural language texts that need to be analyzed and processed to be translated properly. Furthermore, in natural language, ambiguities are usual, where the same word has different meanings depending on the context. It commonly has lexical substitutions also, where, in order not to repeat a word, we refer to it differently. To this, we shall add syntactic dependencies where some entities are bound to others. Lastly, humans can extract and recognize different entities that have a global meaning, like companies, countries, or languages, that should be properly identified in our graph.

Listing 1. Natural language text example. Source: Authors.

```
The La Plata National University (Spanish : Universidad
Nacional de La Plata , UNLP) is a national public re-
search university located in the city of La Plata , capi-
tal of Buenos Aires Province, Argentina.
```

Listing 2. RDF graph for text in Listing 1. Source: Authors.

```
@prefix ex: http://lifia.ar/ontology/ .
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns .
ex: LaPlataNationalUniversity  rdf: type ex: University;
  ex: hasName "La Plata National University"@en ;
  ex: hasName "Universidad Nacional de La Plata"@es ;
  ex: isLocatedIn ex: LaPlataCity ;
  ex: isLocatedIn ex: BuenosAiresProvince ;
  ex: isLocatedIn ex: Argentina ;
  ex: isPublic "true"^^ xsd: boolean ;
  ex: isResearchUniversity "true"^^ xsd: boolean .
ex: LaPlataCity rdf: type ex: City ;
  ex: isCapitalOf ex: BuenosAiresProvince .
ex: BuenosAiresProvince rdf: type ex: Province ;
  ex: isLocatedIn ex: Argentina .
ex: Argentina rdf: type ex: Country .
```

```
ex: University rdf: type rdf: Class .
ex: City rdf: type rdf: Class .
ex: Province rdf: type rdf: Class .
ex: Country rdf: type rdf: Class .
```

In this context, the translation from text to knowledge graph automatically is a desired goal. However, the mechanism used to do it must have the ability to process natural language texts as it must recognize the mentioned characteristics of these texts to express them in the correct RDF format. Several approaches are capable of doing that using regular NLP tools, using DBpedia tools as DBPedia Spotlights in the NER and NEL detection [5] or using the Web LOD cloud [6]. In this article, we analyze the use of a Large Language Model to analyze the natural language. Particularly, this article uses Chat GPT as it is innovative natural language processing. This, added to a detailed prompt to give context to the query, makes Chat GPT a very interesting tool for this task.

A previous open source project named Graph GPT interacts with Chat GPT through its API to build knowledge graphs based on unstructured natural language texts using a prompt. For the graph it maintains an array for entities or nodes and other for relationships or edges. The obtained graph can be extended by writing more information and once it is done it can be stored as a png image. The state of the graph is entirely maintained within the browser and it is not sent in the prompt as, how Graph GPT creator explains, it would make the prompt more expensive and bigger at each query.

This article presents an extension of Graph GPT with different approaches in the type of output and the interaction with the generated graph. With RDF Graph GPT we use Chat GPT to translate natural language text into an RDF turtle format. This makes the first difference with Graph GPT as the Chat GPT prompt won't ask for an extraction of entities and relationships but for a translation to a specific format that requires more processing and the recognition of many aspects in the text, taking advantage of Chat GPT capacities. This RDF is saved in a file, which gives us a lot of opportunities to work with it; we don't need to maintain any structure in order to generate or extend our graphs, the RDF file itself allows us to do these things only by writing or recovering it. Throughout this paper, we will present the RDF Graph GPT operation in detail.

The following sections are organized as follows. In Section 2 the related work is presented. In section 3 we find the methodology, while in Section 4 the text to knowledge graph process is described. In Section 5 the approach is presented. Lastly, in Section 6 the evaluation is presented, in Section 7 we discuss the limitations of the tool; and Section 8 shares conclusions and future work.

2 Related work

There is a long history of attempts to automatically generate knowledge graphs. The one that prompted this project was Graph GPT, which, built as a toy over a weekend, may not be very powerful, but it introduces the idea of using Chat GPT to generate

knowledge graphs from unstructured natural language. It also makes good use of prompt structuring, with the idea of stateless prompts.

As we mentioned in Section 1, Graph GPT maintains a JSON object with the nodes and edges of the actual graph, ones that keeps updating any time that more data is added. One of the limitations of Graph GPT is that it runs entirely on the browser, preventing any kind of saving for the structured data we obtain. The only way to save the graph is as a png image, but that wouldn't allow us to keep extending it after. Other of its limitations is that it doesn't allow more than one relationship between two nodes and that sometimes when the same node is mentioned with a tiny difference, it would take it as a new node.

These issues are resolved with RDF Graph GPT by improving the detailing of the prompt and the way of maintaining the state of the graph. The identification of the nodes is also solved with RDF terms that not only make our graph more readable but also compatible with any other knowledge graph on the web. This means that the same resource will be labeled the same way in our graph and in any other graph.

There are other related works in the matter of knowledge graphs and information extraction with LLMs. A systematic analysis of the KB completion potential of GPT, where we focus on high precision [7]. We meticulously design an open-world evaluation set incorporated in INSTRUCTOPENWIKI to assess Open-world IE capabilities thoroughly, focusing on the generalization of unseen instructions and out-of-ontology entities [8]. Extensive experiments to evaluate the performance, time, and cost-effectiveness of 3 different GPT-3 based data annotation approaches for both sequence- and token-level NLP tasks [9]. This last one also introduces the idea of having a Prompt-Guided Unlabeled Data Annotation that we adopt for our prompts.

3 Methodology

The study introduced in this article carried out the Design Science Research methodology [10], which is based on the creation of meaningful artifacts in the resolution of particular problems [11].

The methodology proposes five steps: (1) Awareness of Problem; (2) Suggestion; (3) Development; (4) Evaluation; and (5) Conclusion. This article focuses on the problem of (1) generation of knowledge graphs from text using LLM. An analysis of Knowledge Graphs in RDF language and the activities needs of the mapping are described below, then RDF Graph GPT is the proposed artifact to solve the problem. Finally, RDF Graph GPT is evaluated with different types of text prompts, and experts analyze the results.

4 Text to Knowledge Graph

The translation of unstructured text to a knowledge graph is lead by a natural language processing (NLP) group of techniques, each corresponding to an aspect of natural

language texts. These are, the entity recognition (NER), the entity linking (NEL), the relationship recognition (REN) and the relationship linking (REL), that we will discuss in subsections 4.1, 4.2, 4.3 and 4.4 respectively. The following are the steps needed in order to obtain good results in the translation, the ones that we desire our tool could do well. Once they are done, we will have a data structure with identified entities and relations, basically, a knowledge graph, where the nodes will be our entities and the edges will be our relationships.

4.1 Entity Recognition

One of the natural language texts characteristics is that, when we read them, we instantly recognize things that have significance for us without the need of having an explanation. This means that we can implicitly put them in context. Named entity recognition or NER is the process by which we can recognize and categorize entities in a text that have a global meaning. This entails the identification of the tokens that represent entities and their subsequent classification in known subgroups like country, organization, language, etc.

4.2 Entity Linking

Another thing to mention about natural language texts is the appearances of words that are written exactly the same but have different meanings depending on the context. To disambiguate this or any other possible confusion, we use named entity linking or NEL, that, as its name suggests, links all entities with a unique identifier from some knowledge base like DBpedia or Wikidata.

4.3 Relationship Recognition

In texts, we will find connectors between the different entities. There is the need to recognize these relationships too, as they will be the connectors of our graph nodes. To achieve this, we use relationship extraction or REN, not only to recognize them but also to categorize them, as they could be of different kinds like a class, a property, etc.

4.4 Relationship Linking

Lastly, we shall do with relationships like with entities; globally identify them in the context of the language. For that, we use relationship linking or REL

5 Approach

5.1 RDF Graph GPT functionalities

A visualization of RDF Graph GPT is provided by Fig. 1. We can split its functionalities in the subsequent parts:

Translation. Initially, this app translates natural language into RDF format. To do so, it is necessary to count on a translation tool that recognizes the entities in the text. Especially with RDF the tool should not only recognize them but also categorize these entities to use a proper term for them, as they are usually existent wikidata terms. The terms are the tool that RDF gives us to identify our resources globally. RDF describes the resources in a subject-predicate-object structure, these three elements, when they are put together, is what we call a triplet in RDF. So, in addition to the entities, we need a tool that recognizes the relations between them, as long as the objects of the sentence, in order to form all the triplets.

Once that is settled, there are other details that the tool needs to acknowledge. When it comes to the identification of a resource, the tool may use different kinds of terms depending on the resources semantic. If it is a unique resource it should use an IRI as its ID, so it's globally identified in one and only one way, but if it's a name, an address, or some kind of human readable information, it should use Literals to identify them, as we want these resources to maintain its string state. Lastly if there is need to represent a resource that is actually unknown, for example the author of a book, it should resort to a blank node. On the other hand, in natural language is common to find lists as the object of the sentence; in those cases, the tool should be aware of that, as there is a certain way to express the lists in RDF format [12]. The chosen tool for the translation is Chat GPT with which we interact through its API using a prompt.

Prompt flexibility. Although RDF Graph GPT's user interface doesn't allow any interaction with the prompt, in the visualization of it as a framework it's a very easy task to do. We can add as many prompts as we want and use any of them to do the request to the API, thus to consult Chat GPT. This is a very interesting aspect as it could be helpful to try different kinds of prompts and see the obtained results depending on which we use. By changing the prompt, for example, we could change the instructions that we send to the model, or the example we use to present our desired output, as, how we were able to see, the improvement in any of the aspects of the prompt means a direct improvement on the generated graph. In order to do this, we need to create a .prompt file, then write on it the desire prompt, always adding a variable that we are going to replace with the text that we want to translate. Then, when doing the fetch to the API we have to take that prompt with the text injected and send it.

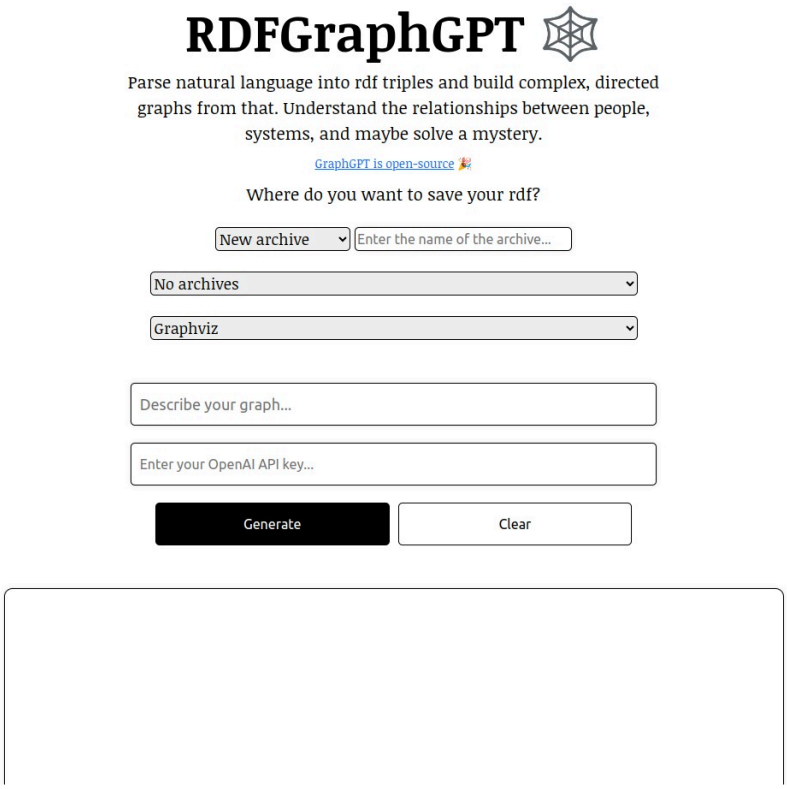

Fig. 1. Visualization of RDF Graph GPT. Source: Authors.

Graphic flexibility. In RDF Graph GPT, our RDF graph will be visually graphed through nodes and edges. The platform offers a variety of possibilities to interact with the obtained graph. For example, in addition to graph an existent file if it is in RDF format or create one in a new file, one could extend an existent file by adding more information to it, which will be displayed in a new graph.

Furthermore, one can choose the graph framework between Graphviz and Cytoscape. While Graphviz draws a more visual and simple graph, the last one allows the user to customize the nodes and edges names, in a visual approach, as it will not change the file content.

RDF improvement. Lastly, RDF Graph GPT allows the improvement of the result in two different ways. One is with the help of Chat GPT and the other is on hands of the user. The first one uses a prompt to request the API to swap some IRIs for the ones that the user enters. The second one lets the user change any part of the RDF resultant code, either change the IRIs or triplets or add new ones. This change, in contradistinction to the one Cytoscape offers, will change the involved file content; as long as the graph, as it will generate a new one.

5.2 RDF Graph GPT architecture

An overview of the RDF Graph GPT architecture is provided by Fig. 2. The architectonic elements that conform this app are mainly the react app, the node server that continually interacts with the file system and finally the API that we consult and which represents our communication with Chat GPT.

The process begins when the user inserts a natural language text and an API key, and requests to generate the graph. On that point, the app takes the general prompt and inserts the input text in it, then the app fetches the API with the entire prompt and the API key. The response comes in an array in which we find some options, such as the reason of the response to end. Also, one of the options and the most important in this case, is the text field, as in it we find the RDF.

Once it has the response RDF the app checks the location where the user wants to save it. Could be, as we said before, a new or an existent file. With that information, the app reaches the server, sending the RDF and the name of the file. It is the server which saves the information in the corresponding location. Meanwhile, the app checks which of the two graph frameworks were chosen and depending on that, it translates the RDF into a DOT or JSON format and graph the entire RDF with it. To clarify, if the user is extending an existent file, the app will recover the previous RDF and add the new extension so all of it is graphed.

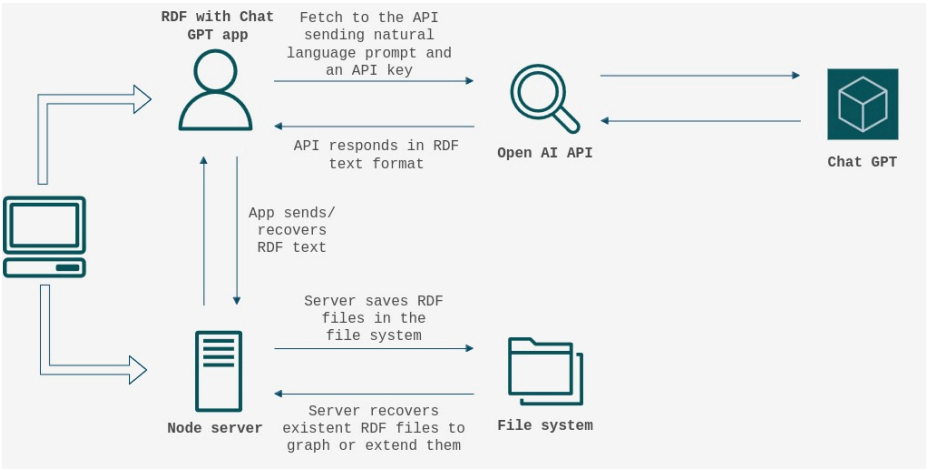

Fig. 2. Architecture of RDF Graph GPT. Source: Authors.

5.3 RDF Graph GPT prompts

When it comes to interacting with an AI, one of the most important things are the prompts, as they are the way for us to ensure we reach an accurate and coherent answer.

Listing 3. Guided prompt with abbreviated example. Source: Authors.

```
Please represent the following natural language input as
an RDF turtle format. If necessary, clarify any ambiguous
entities or relationships. Avoid using incorrect wikidata
ids or dbpedia ids. Use several vocabularies such as wik-
idata, dbpedia, schema.org, yago, foaf and other type of
rdf vocabularies that appears on the web. Make the de-
scription complete and correct! For wikidata you can
search for IRIs in https://www.wikidata.org/?uselang=en
and for dbpedia from https://www.dbpedia.org. For ele-
ments without a DBpedia or wikidata URI use the follow-
ing: http://lifia.ar/ontology/. Please use all the tools
that the turtle format allows, as the collections and
classes. When the response ends, please put the string
“eof”, so i can know if i have all the information.
Example:
Input:
“The apartment block with address 146, 5th Avenue, New
York, US, has 24 floors and 120 apartments. It was con-
structed in the year 1976. The second floor of the apart-
ment block has 5 apartments: 201, 202, 203, 204 and 205.
. . . ”
Output:
@prefix ex: http://lifia.ar/ontology/ .
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns .
@prefix foaf: http://xmlns.com/foaf/0.1/
ex: ApartmentBlock rdf: type ex: ApartmentBlock ;
  ex: hasAddress “146, 5th Avenue, New York, US” ;
  ex: hasFloors “24”^^xsd: integer ;
  ex: hasApartments “120”^^xsd: integer ;
  ex: wasConstructedInYear “1976”^^xsd: gYear .
  ex: hasFloor ex: SecondFloor .
ex: SecondFloor rdf: type ex: Floor ;
  ex: belongsTo ex: ApartmentBlock ;
  ex: hasApartments (ex: Apartment201, ex: Apartment202,
ex: Apartment203, ex: Apartment204, ex: Apartment205) .
ex: ApartmentBlock rdf: type rdf: Class .
ex: Floor rdf: type rdf: Class .
eof
Input: $prompt
Output:
```

In this app we use a non-chat completion model, which means that the endpoint is expecting a single string prompt instead of a list of messages [13]. An example of the prompt is shown by Listing 3.

The examples turned out to be a powerful tool for the prompts; the tests showed that without them the responses were very varied and not accurate. As a result, the app prompts are guided ones, which structure is made of a brief description of what we want to obtain from the model. After, we find an example of input followed by an example of the perfect output for the given input. Then a variable that will be replaced with the input of the user and lastly a hook that will be the trigger for the model response.

RDF Graph GPT uses two different prompts depending on the action we intend to do. One for creating a new RDF and the other for improving an existent one. Both of them follow the structure we already described. An important thing to consider is that when it comes to extending the RDF Graph, we use the first prompt as if we were creating a new RDF. We do this in order to make the prompt stateless, as we don't need to send the actual state of the RDF in order to ask Chat GPT to extend it, we can just create a new RDF with the new information and then save it in the original file; that way, we are automatically making the RDF bigger, taking advantage of RDF graphs union capacity. The stateless capacity of the prompt is very important given that if we send the actual state of our graph in each request, the request would be not only more expensive but also bigger each time we extend the graph, leading a point where the API will reject our petition because it will exceed the maximum token amount.

6 Evaluation

Given the context and the desired results in the matter of knowledge graphs, we are going to evaluate the resultant RDF graph correctness. As we mentioned in section 4 there are some needed tasks that the translation tool must perform, whose outcomes are of our interest. So, in the resultant RDF graph, we are going to evaluate (1) Correct entities and literals, (2) Correct relationships, (3) Vocabulary reuse, and (4) Usage of a given IRI when there isn't reuse. The correctness of relations and entities has two important aspects, one is the existence of those entities and relations in the natural language text that we use in the prompt, basically, that the resultant graph makes sense; and the other involves the point (3), that is the correct linking, this means that both relations and entities are linked to existent vocabularies in RDF published on the Web. Lastly we will expect that when no reuse is possible, the tool use a given IRI that we will provide or generates a coherent IRI.

We will evaluate these aspects trying different prompts. As we mentioned, our prompt is a guided one, this means that it could be separated in two parts; the training data and the actual text that we want to convert into RDF. This evaluation will consider changing not only the training part but also the type of text that we give to the model.

When it comes to the configuration of the guide part, we will be trying three different prompts. The first is a training part with detailed instructions and a short example (inst-1); the second has also detailed instructions but no example (inst-2); and the third has simplified instructions and a large example (inst-3). We want to analyze how the model reacts to more details in the instructions and to the examples.

On the other hand, for the texts we choose two types of texts fragments. The first is a wikipedia fragment where we know that the entities come from wikidata (txt-1); and the second is a fragment of a book (txt-2), as we want to analyze how the model reacts to entities that may not have a wikidata IRI.

This evaluation will be done by experts that given a text and a type of prompt will analyze the sense of the resultant RDF graph and the correctness of the IRIs. This analysis will have a (1) quantitative and a (2) qualitative aspect. The first one refers to the amount of correctness, this is, given a graph, how many relations and entities are correct considering both meaning and linking and how many are incorrect; and the second are the possible improvements that would lead to better RDF graphs; lastly, having this information we will evaluate if the graphs can be improved by the existent editing functionality.

On the table 1 we can see the different variations of instructions and text fragments with the resultant RDF for each. They are identified by an id that matches with other in the document which link we left below, where these three resources are written, as they would take a lot of lines in this paper.

Table 1. Table with results. Source: Authors.

Prompt	Chosen text	Generated RDF
Inst-1	Txt-1	Res-1
Inst-2	Txt-1	Res-3
Inst-3	Txt-1	Res-5
Inst-1	Txt-2	Res-2
Inst-2	Txt-2	Res-4
Inst-3	Txt-2	Res-6

On the other hand, Table 2 shows the different expected entities by both evaluators for each text fragment.

Table 2. Table with expected entities. Source: Authors.

Text	Expected entities by evaluator 1	Expected entities by evaluator 2
Txt-1	La Plata National University; Universidad Nacional de La Plata; UNLP; University; La Plata; Buenos Aires Province; Argentina; 90000 Literal; 10000 literal; 17; 106; University reform; movement; National University of Cordoba; 1918 literal; Argentine university systems; free university education; autonomy; curricular	Universitary reform; Q1570489 (UNC); Q784171 (UNLP); Q44059 (La Plata); Q44754 (Buenos Aires); Q414 (Argentina); “University reform in Argentina was a .... as well as many others in Latin America.” ; “1918”; “106” ; ”17” ; “10000”; “over 90000”

	flexibility; university extension; Latin America	
Txt-2	Game mechanic; term; rule; set of rules; player action; cause-and-effect relationship; player/s; Action; game status; expected consequence; game; simulation; (novel) mechanics; repurpose; recombine; modify; example (of game mechanic); stealth game mechanic; ability to avoid been seen; penalty; educational games; situations; person's perspective	Stealth-game-mechanic Q1751513 (game mechanic class) "Giving the players the ability to avoid ... another person's perspective."; "The term "game mechanic" ... lots of games.";

6.1 Evaluator 1 – analysis

inst-1, txt-1 (res-1). The generated entities match with the expected ones except for two of them. Besides, the result shows some extra entities. When it comes to the correctness of these entities 20 of them are correct and 1 is incorrect, it generates clas-ses with semantic problems. The relations are correct; it generated 58 syntactic and semantic correct triplets. There was zero vocabulary reuse founded, every IRI was new except for rdf:type.

inst-2, txt-1 (res-3). 3 expected entities where detected but with bad IRIs, 11 of them weren't detected and 7 new entities were founded of which 6 were incorrect and 1 was correct. For the relationships 24 triplets with correct relations were founded in a total of 49 triplets. When it comes to vocabulary reuse, on the majority of the triplets the tool intended to use schema or wikidata IRIs but they are not correct; it generates properties that doesn't exist or uses wikidata IRIs that match with elements with no sense in the context. In this case we didn't find much sense on the results, the IRIs don't match with the text and the descriptions don't seem right.

inst-3, txt-1 (res-5). In this combination, 19 entities were correctly detected, 3 weren't detected and 5 new entities were generated, all of them correct. For relationships, 47 triplets make sense as they use the base URI as reference but there are some class definition triplets that are conceptually incorrect. There wasn't vocabulary reuse except for rdf:type and rdf:Class, all the rest triplets used the base IRI generating or inventing possible relations or classes.

inst-1, txt-2 (res-2). Only 5 of the expected entities where found, Game mechanics, game, simulation, stealth game mechanic and mechanic type. All of them are correct

but there are a lot missing. When it comes to the relationships, 12 of the triplets were correct, 4 incorrect. The incorrect have the structure “Game rdf:type Game” which is incorrect because it says that an entity is of the type of itself. Except rdf:type and rdf:Class there isn’t vocabulary reuse, the properties are created using the prefix ex:.

inst-2, txt-2 (res-4). 6 of the generated entities weren’t expected but we can take them as correct because they are bounded to the expected ones. 33 triplets were generated but with questionable correctness as the class definition triplets don’t use the prefix rdf but owl which is incorrect on the context; and when it comes to the comments that the tool adds to the class definition, they are invented, not extracted from the text. As to the vocabulary reuse, the analysis is similar from the ones above, it adds the rfds:label and rfds:comment. The RDF looks coherent but not correct as it incorporates elements that don’t come from the text.

inst-3, txt-2 (res-6). 14 of the expected entities and 6 extra entities were generated, from the not expected ones, one is correct and in the text, one is correct but wasn’t in the text and the other four were incorrect. As for the relationships, 41 correct triplets were generated. The model didn’t reused almost any vocabulary, just the rdf:type label. When it comes to the coherence of the RDF this result is better than the one above but it is important to observe that the model takes comments from other sources outside the text
ddd

6.2 Evaluator 2 – analysis

inst-1, txt-1 (res-1). All of the expected entities except three were generated; but not with the wikidata syntax as it was expected. Moreover the model generated 7 new entities that weren’t expected. Some of these weren’t directly on the text like “Cordoba” or the seek changes of the university reform, which have sense, but it’s interesting that the model inferred them from the text. As for the relationships, all of them are considered correct. The model was coherent on the generation and usage of entities and relationships.

inst-2, txt-1 (res-3). The generated entities are questionable, the tool intended to use wikidata IRIs but they were all wrong; besides, to model that UNLP has 17 departments it generated a collection with a different id for each one of them. On the other hand, it correctly used schema.org relations. Beyond that, the tool didn’t reuse vocabulary; it redefines identifiers from the space-name of wikidata without considering that they already exist.

int-3, txt-1 (res-5). All of the generated entities were correct and expected, they make sense for the context. The relationships that the model defined were correct. However the model did not use vocabulary reuse.

int-1, txt2 (res-2). Just two of the generated entities were correct, another was questionable, and the two others were incorrect. The model initialized some classes as instances of themselves, which is incorrect. In summary, the model didn't use existent vocabulary but it was consistent with what it defined.

7 Limitations

Although Chat GPT is quite good at natural language processing, the model presents some limitations when it comes to our task. What we need is a translation, not between languages but between data formats. NLP plays a very important role in this translation but there are other things to take into account. In our case, these things are the specific RDF turtle syntax and the usage of existing IRIs. The fact that Chat GPT can't be pretrained makes us unable to impose the context that we want to the model, so we usually find some syntax errors or invented IRIs on the resultant RDFs. The first limitation can be avoided by providing the model a prompt that includes an example in which the syntax is shown or using a fine-tuned model; but the second comprise a higher degree of difficulty as we can't neither show all the possible IRIs in an example or fine-tunning task, nor train the model to generate them. We think this problem might be solved with Chat GPT 4, which has access to the internet, and, therefore, to the different defined ontologies. On the other hand, another solution for this problem is to train an open-source LLM like LLaMA that we can shape for our specific task.

8 Conclusions and Future work

Throughout this paper we have presented RDF Graph GPT as a tool to translate natural language into knowledge graphs, particularly using RDF format. We have gone over the importance of knowledge graphs in order to provide semantics to the data, as long as the promising and ongoing tool that the IA represents for us in this context. This app, then, puts the focus on the IA as a tool on the matter of developing powerful tools for this translation, and demonstrates that tools like Chat GPT can bring us up-and-coming results that are in continuous improvement.

Based on the analysis made by experts, we have seen that this tool works quite good when it comes to processing natural language, especially when we provide it with a detailed guided prompt with an example; this was proved by the amount of correct entities and relationships that the tool could recognize using inst-1. However, the results where different on the matter of vocabulary reuse. Instead of finding an existent RDF IRI, it invents it using the correct structure or finds a real IRI that makes no sense for the entity. This and other details may be fixed with GPT-4 that has access to the web. In the meanwhile, it is a task that we are able to do with the improvement functionalities that RDF Graph GPT provides, although they require of our intervention and knowledge.

On the other hand, another conclusion that the evaluation gives us is that the results are significantly better when the text that we request to translate comes from the Wikipedia. We observed that with txt-1 the inst-1 prompt recognized almost every expected entity and all of the relationships were correct. On the contrary, using that same instructions with txt-2 a lot of entities were missing, although the ones that it indeed found were correct.

Beyond this analysis, we have to take into account that it was made looking at a particular LLM results. There are new models of GPT and other LLMs with powerful qualities that we consider will be of great importance on the matter of translating natural language texts into knowledge graphs.

As a future work we would like to make this same analysis but using other LLMs or Chat GPT-4 in order to find the most compatible model for our task.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Sci. Am. 284(5), 34–43 (2001)
2. Hogan, A.: Web of Data. In: The Web of Data, pp. 15–57. Springer, Cham (2020). doi: 10.1007/978-3-030-51580-5
3. Jain, N., Sierra-Múnera, A., Streit, J., Thormeyer, S., Schmidt, P., Lomaeva, M., Krestel, R.: Generating Domain-Specific Knowledge Graphs: Challenges with Open Information Extraction. Text2KG 2022: International Workshop on Knowledge Graph Generation from Text. CEUR Workshop Proceedings (2022). Available online: https://ceur-ws.org/Vol-3184/TEXT2KG_Paper_4.pdf
4. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. de, Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.-C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge Graphs. Synthesis Lectures on Data, Semantics, and Knowledge. 12, 1–257 (2021). https://doi.org/10.2200/S01125ED1V01Y202109DSK022.
5. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems. pp. 121–124. Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2506182.2506198.
6. Rincon-Yanez, D., Senatore, S.: FAIR Knowledge Graph Construction from Text, an Approach Applied to Fictional Novels. In: TEXT2KG/MK@ ESWC. pp. 94–108 (2022).
7. Keming, L., Xiaoman, P., Kaiqiang, S., Hongming, Z., Dong, Y., Jianshu, C.: PIVOINE: Instruction Tuning for Open-World Information Extraction. arXiv preprint. https://arxiv.org/pdf/2305.14898.pdf (2023)
8. Veseli, B., Razniewski, S., Kalo, J.C., Weikum, G.: Evaluating the Knowledge Base Completion Potential of GPT. arXiv preprint. https://arxiv.org/pdf/2310.14771.pdf (2023)
9. Bosheng, D., et al.: Is GPT-3 a Good Data Annotator? arXiv preprint. https://arxiv.org/pdf/2212.10450.pdf (2022)
10. Vaishnavi, V.K., Kuechler, W.: Design Science Research Methods and Patterns: Innovating Information and Communication Technology, 2nd edn. CRC Press (2015)
11. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. Manag. Inf. Syst. Q. 28(1), 75–105 (2004)

12. Hogan, A.: Resource Description Framework. In: The Web of Data, pp. 59–109. Springer, Cham (2020). doi: 10.1007/978-3-030-51580-5_3
13. OpenAI: Overview - OpenAI API. https://platform.openai.com (2020). Accessed: Mar 2024, last accessed 2023/10/25