Advanced Planning and Scheduling with Semantic Knowledge Graphs and LLMs

Nenad Petrovic[1,†], Milorad Tosic[1,*,†]

[1] University of Nis, Faculty of Electronic Engineering, Aleksandra Medvedeva 4, 18104 Nis, Serbia

Abstract

In this paper, ontologies-driven approach is proposed to facilitate integration of freeform textual descriptions as inputs to an Advanced Planning and Scheduling (APS) solution. The approach uses Large Language Model (LLM) for automated Semantic Knowledge Graph (SKG) construction starting from an input freeform text. It is complemented by a set of ontologies as well as a corresponding knowledge graphs. If size of ontologies is larger than the context of commonly used LLM solutions, Retrieval Augmented Generated (RAG) method is adopted, specifically Retrieve and Re-Rank (RRR) procedure. Traditionally, modifications of ontology in semantics-driven intelligent systems usually result in need to make additional source code changes and updates of the supportive tools, while in the proposed approach only change of input ontology and example knowledge graph are enough. Based on results of the presented case study, we conclude that synergy of LLMs and RAG exhibits strong improvement potential in cases when semantic annotation is critical for system performances. Further work is required to explore the potential in more details, including reduction of costs and adaptability improvement of APS adoption in challenging environments such as smart industry and discrete manufacturing.

Keywords

Ontology, Semantic Knowledge Graph, LLM, Planning, Scheduling

1. Introduction

Advanced Planning and Scheduling (APS) is described as "any computer program that uses advanced mathematical algorithms or logic to perform optimization or simulation on finite capacity scheduling" [1]. Operation in a modern digital enterprise, such as is the case with discrete manufacturing, is becoming increasingly complex, especially considering scalability and diversity of involved production resources together with high market requirements dynamics. As a consequence, APS becomes more and more important while quite challenging in the state-of-art Industry 4.0 environments [2, 3]. Recent advances in Artificial Intelligence (AI) offer a promising approach to addressing the challenge.

The development process of AI can be conceptualized along the following three dimensions [4]: 1) *Perceptual intelligence* means that a machine has the basic abilities of vision, hearing, touch, etc., corresponding to human senses. 2) *Cognitive intelligence* is a higher-level ability of induction, reasoning and acquisition of knowledge. It is inspired by cognitive science, brain science, and brain-like intelligence to endow machines with thinking logic and cognitive ability similar to human beings. 3) *Decision intelligence* requires the use of applied data science, social science, decision theory, and managerial science to expand data science, so as to make decisions that are optimal with respect to commonly accepted shared knowledge. Even if a machine has the abilities of perception and cognition it would not be enough for it to make optimal decisions in complex environments involving human beings, such as industrial manufacturing, medicine, etc. APS, and intelligent decision support systems in general, can be considered as decision intelligence. Perceptual intelligence, on the other

[1]*Proceedings of the Joint Ontology Workshops (JOWO) - Episode XI: The Sicilian Summer under the Etna, co-located with the 15th International Conference on Formal Ontology in Information Systems (FOIS 2025), September 8-9, 2025, Catania, Italy,*

[*] Corresponding author.

[†] These authors contributed equally.

nenad.petrovic@elfak.ni.ac.rs (N. Petrovic); milorad.tosic@elfak.ni.ac.rs (M. Tosic)

0000-0003-2264-7369 (N. Petrovic); 0000-0001-8142-5788 (M. Tosic)

side, has generated a lot of hype in recent years by great advancements in deep learning, generative intelligence and Large Language Models (LLM). However, cohesive role of cognitive intelligence, necessary for development of reliable and explainable solutions, has not been fully understood yet. In this paper, we leverage ontology-driven APS solution and propose Large Language Models (LLMs) in order to enable planning based on inputs given in a form of textual descriptions that are more convenient to end-users. Also, end-users do not necessarily need specific expert knowledge necessary for mastering set of ontologies and semantic data used in the APS. In this way, high potential of adopting ontologies for implementation of the cognitive layer is proven.

In order to tackle APS in manufacturing domain, many mathematical, optimization-based and formally grounded approaches have been proposed [2, 3]. However, all of them rely on models, notations and representations (such as linear equations of mixed integer linear program) [2, 3, 5] whose adoption requires additional cognitive load in order to apply them successfully in practice. One of the promising solutions that is aiming to reduce cost of the cognitive load combines ontology-driven approach to semantic knowledge graph representation and optimization-based planning. Semantic Knowledge Graph (SKG) is representation of a specific knowledge domain using ontologies for conceptualization. In SKG, resources (nodes) are connected by relationships (edges), contextualized by semantics in order to integrate and reason about knowledge in a way that both humans and machines can understand and use. While cognitive load of abstract modeling and pure mathematical representation is eliminated thanks to ontology-based domain knowledge representation, adoption of ontologies for semantic annotation of planning-related information (including orders, production machines and other resources) might still impose additional costly efforts and cognitive load when it comes to understanding key concepts and their relations [6].

AI-enabled approaches have opened many new horizons for innovation, as well as enhancement of the existing applications across different domains, including industry and manufacturing [7]. The cognitive load of using the underlying ontologies is still much higher than in the case where end-user enter simple natural language freeform text. In this paper, LLM-based approach for automated ontology-driven semantic knowledge graph construction from freeform user-provided textual input is proposed. In addition, the paper tackles the issue of leveraging large amount of textual data representing the set of adopted ontologies. Size of the input text is identified as one of the main challenges affecting the performance of the LLM results' accuracy, as well as one of barriers for realization and adoption of such scenarios in practice. Retrieval Augmented Generated (RAG) method is adopted, particularly Retrieve and Re-Rank (RRR) process, when content of the set of ontologies is larger than the context of commonly used LLM solutions. Additionally, we evaluate variations of prompt engineering techniques [8], with the focus on the so-called In-Context Learning (ICL) [9] prompting strategy, where task examples are integrated directly within the prompt itself. The main advantage of the approach is that pre-trained LLMs can perform new tasks without requiring additional fine-tuning, which is time and resources intensive procedure.

2. Methodology

Considering the limited context size of LLMs, direct use of textual serialization of planning related ontologies as a context within prompts is not practically useful due to large volume of data contained. This naïve approach would either lead to huge costs resulting from token consumption (each time we have huge ontology as part of the prompt context) as well as hallucinations in the case when the context is arbitrarily cut. Therefore, additional strategy is needed for preprocessing of textual content of the target domain ontologies before it is used for automated LLM-driven semantic annotation of user-defined freeform text. For the preprocessing, Retrieve and Re-Rank [10] approach is adopted for context construction in the ICL-based question answering strategy [11] as a robust method for information retrieval in question-answering systems [8]. A high-level overview of proposed RRR based approach, referred to as RRR4ICL, is given in Figure 1.

The proposed approach of context construction consists of the following three main phases:

- *Preprocessing*: transform input textual files into format suitable for next phases (such as splitting document into chunks).
- *Retrieve*: Initial retrieval of a broad set of potentially relevant parts of text (chunks).
- *Re-Rank*: Re-ranking to isolate the most relevant subset of chunks.

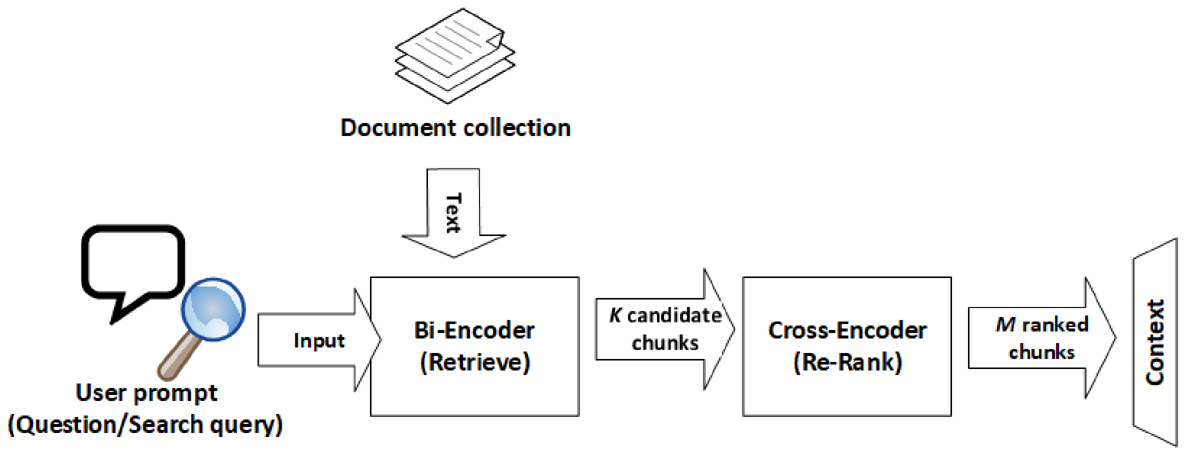

Figure 1: Retrieve and Re-Rank for In-Context Learning (RRR4ICL) approach workflow overview, K>M.

There are the following two types of inputs in retrieve step: *a) document collection* – textual content used as basis for user prompt answering - in our case, it will be either a single textual file composed by concatenation of content of the specific domain ontologies, or a representative SKG. *b) user prompt* – textual input provided by user that will be used as basis for retrieval of relevant information from document collection. In general, there are the following two possibilities for what user prompt can represent: *i)* question about the documents' content that user expects to get answer for, or *ii)* search query where user expects to retrieve relevant documents.

As the recommended option for general-purpose text preprocessing, *RecursiveCharacterTextSplitter* [12] is adopted for creation of document chunks. It uses a prioritized list of characters to determine where to split the text, attempting each in order until the resulting chunks are within the desired size. The default character list is ["\n\n", "\n", " ", ""], which means it tries to preserve larger semantic units—like paragraphs, then sentences, then words, as much as possible. Therefore we can sum up that splitting method is based on a specified list of characters, while chunk size metric is number of characters. There are two relevant parameters of text splitter responsible for creation of text chunks based on given document that need to be specified:

- *chunk_size* - maximum number of characters or tokens allowed in a single chunk
- *chunk_overlap* - number of characters or tokens shared between consecutive chunks. Overlapping ensures that important context is not lost when dividing the text into smaller parts.

In our case, we use *chunk_size* equal to 600 characters, while *chunk_overlap* is up to 200 characters, based on empirical estimation.

After the preprocessing phase, the second phase begins with semantic search using the *Bi-Encoder* model [13]. Given a search query, a large set of potentially relevant text segments (or chunks) is generated by retrieval. It is performed using dense retrieval, specifically a *SentenceTransformer*[2]-based Bi-Encoder in our case. However, this method can sometimes return results that are only loosely related to the query. To address this, the third phase is introduced: a re-ranking model based on a *CrossEncoder*[3], which evaluates and scores the relevance of each candidate segment more precisely. The final output is a ranked list of results optimized for relevance. The dense retrieval approach

[2] https://sbert.net/docs/quickstart.html#sentence-transformer
[3] https://www.sbert.net/examples/cross_encoder/applications/README.html

leverages semantic search [14], mapping both the query and documents into a shared vector space to retrieve the closest matches. This method surpasses traditional lexical search by recognizing synonyms, acronyms, and semantically similar terms [14].

Two-stage retrieval setup is adopted that aims to balance between efficiency and accuracy. In the first stage, sentences and paragraphs are converted into 384-dimensional dense vectors optimized for semantic similarity calculation task using *multi-qa-MiniLM-L6-cos-v1*[4] model. To complement the *Bi-Encoder*, the *cross-encoder/ms-marco-MiniLM-L-6-v2*[5] model is incorporated in the second stage. The overall process including context construction and prompt execution is broken down into steps as given in Table 1.

Applying this approach, only relevant concept definitions and relations are extracted from the large volume of text (a single ontology/knowledge graph file maybe large or the set of data files may contain large number of them), which will be further used as query context. The goal of queries in our case could be either answering the questions about the concepts, their properties and relations or creation of semantic knowledge graph based on freeform textual input with respect to the given set of ontologies. Without such an approach, a naïve solution to avoid frequent hallucinations would be to include full text of the serialized set of ontologies or knowledge graphs as part of the prompt context, which would either lead to exceeding the context size limits of most LLMs or increased token consumption (slower execution time and greater costs as well) even if the full set of ontologies can fit the selected model's context size. The aim of the applied approach is to overcome context size limitations and increase prompting efficiency in the same time.

Table 1
RRR4ICL steps

```
Input: query, document
Steps:
1. Split the document into chunks
2. Bi-Encoder rapidly computes similarity scores (ranging from 0 to 1) between the input
   query and all document chunks.
3. Select the top k chunks (where k = 32 in our case) based on the similarity scores.
4. Apply Cross-Encoder:
     4.1. Each of the top k chunks, along with the original query, is passed through the
          Cross-Encoder to refine the ranking.
     4.2. Top m results (where m = 3 in our case) are selected as the final context to be used
          for prompt execution

Output 1:  context - top m chunks combined
5. prompt:=parametrizePrompt(query, context)
6. answer:=executePrompt(prompt)

Output 2: answer – response generated by LLM
```

3. Implementation Architecture

Workflow of the proposed solution for ontology-driven APS aided by LLM agent is given in Figure 2.

[4] https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1
[5] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2

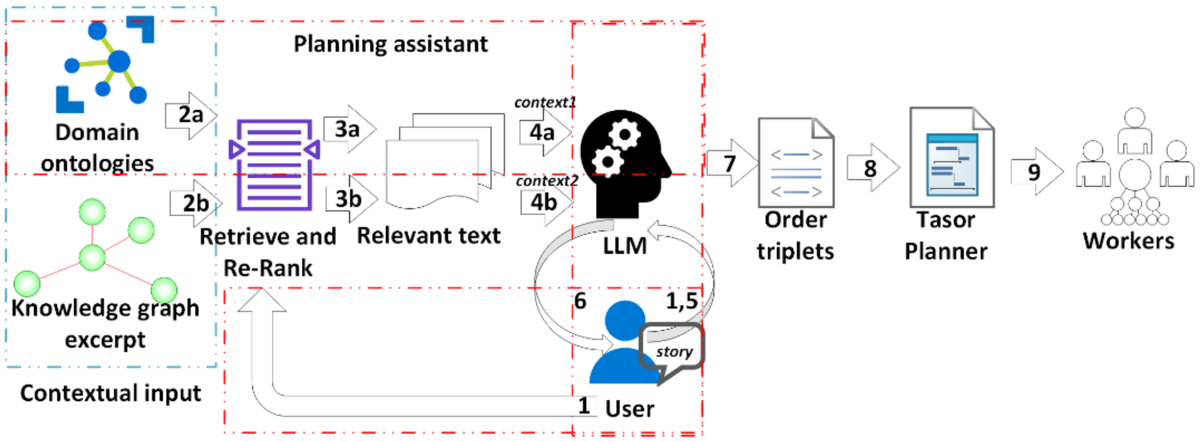

Figure 2: LLM-aided planning workflow: 1,5-User input {story} 2a-RDF schema 2b-RDF knowledge graph 3a-Ontolgy chunks 3b-Knowledge graph excerpt chunks 4a-Context: ontology excerpt {context1} 4b-Context: knowledge graph template {context2} 6-Intermediary result 7-Triplets 8-Order definition 9-Generated work plan.

In the first step, user describes what is ordered together with related resources (such as machines and employees) as a freeform text. This user-provided text, {story}, is leveraged as input of RRR process against the contextual input (Figure 2). As outcome of the RRR process, the most relevant excerpts of the ontology collection based on the user description are extracted (denoted as 4a – context1 in Figure 2), so it can be further used as context of prompt to LLM service. On the other side, in order to further make the results more accurate, a representative excerpt of knowledge graph is taken into account in order to ensure that LLM will hit the right term names without hallucinations (denoted as 4b-context2 in Figure 2). While RRR related steps are executed on host machine, we rely on GPT-4o as LLM service for answering user prompts. In this case, the prompt template containing user-provided {story} has the following structure and is executed in step 5:

***Prompt 1**: "Create set of RDF triplets for semantic knowledge graph in XML with respect to given ontologies: {context1 – ontology excerpt} and example graph {context2 – knowledge graph template} based on user story: {story}"*

Here, *{context1} and {context2}* are outputs of RRR process, while *{story}* is user-defined freeform text describing the desired order or questions related to the underlying order-related concepts (Figure 2).

The following prompt is used for updates and extensions of given knowledge graph with respect to user story:

***Prompt 2**: "Update given semantic knowledge graph in {graph} based on user story: {story}"*

In addition to order description, it is possible for user to include other factors in the story as well, such as employees and machines involved. Output would be a set of triplets describing the order that can be inserted into the semantic knowledge graph, which is further used as input to the APS. The workflow could be enhanced in an interactive direction by enabling user to see intermediary results and provide additional textual feedback to refine it. In this case, *prompt1* is extended taking into account the current result, as well as its beginning is modified as follows: "*Update set of RDF triplets for {current result}"*. The final output of the overall workflow from Figure 2 is manufacturing plan produced by the APS, considering the previously created order relying on LLM agent.

On the other side, the subset of components bounded by red line in Figure 2 provides capabilities of LLM planning assistant that is able to answer the user's questions about concepts from planning ontologies, their relations and properties. In this case, only *context1* based on ontology excerpt is used, as it can be seen in Figure 2. The aim of such tool is to provide "hints" while describing order and help users understand the key aspects which should be covered by the freeform text of the provided story. Structure of the prompt is:

***Prompt 3**: "Answer the question about ontology: {story}, based on given excerpt: {context1 – ontology excerpt}"*

3.1. Deployment overview

For implementation of the previously described workflow Python is adopted, with LangChain[6] and Ollama[7]. LangChain refers to an open-source framework whose aim is to aid developers build applications relying on LLMs in more effective and efficient manner, by providing set of tools and abstractions. Main concepts of LangChain can be identified as [15]: 1) *chains* - sequences of calls to an LLM and other tools as a logic flow that combines multiple steps into an unified process; 2) *agentic approach* - allow LLMs to make decisions dynamically, based on current context; 3) *memory* - keeps context between calls (such as conversation history); 4) *document loaders and retrievers* - query information from files, databases, websites, large textual documents; 5) convenient tool Integration - easily plugs into web browser; 6) *wide LLM support* - including OpenAI, Anthropic, Hugging Face and many others.

On the other side, Ollama is an open-source platform that enables running and management of LLMs directly on local machine [16]. It simplifies the deployment of open-source models like Llama 3, Mistral, Phi, Gemma, and DeepSeek, allowing us to interact with them through a terminal or API without relying on cloud services. Overview of the possible deployment is given in Figure 3.

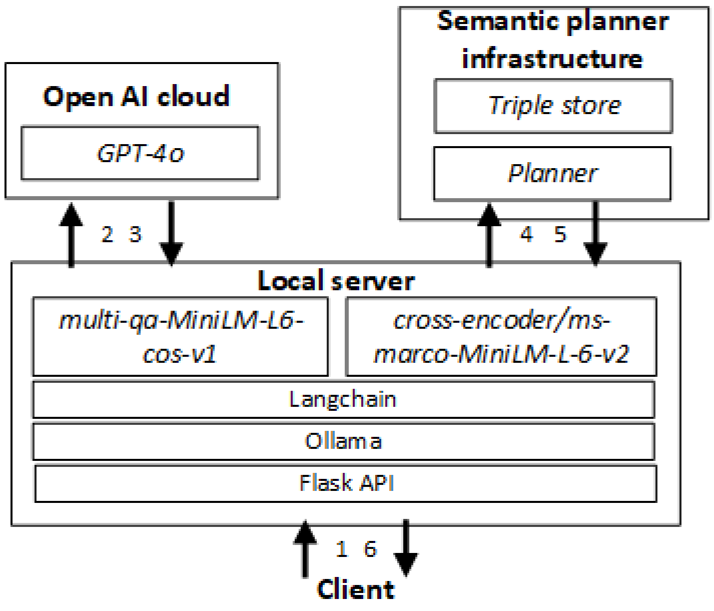
Open AI cloud
GPT-4o
Semantic planner infrastructure
Triple store
Planner
2 3
4 5
Local server
multi-qa-MiniLM-L6-cos-v1
cross-encoder/ms-marco-MiniLM-L-6-v2
Langchain
Ollama
Flask API
1 6
Client

Figure 3. Deployment overview: 1-HTTP request containing user story; 2-Prompt to GPT-4o; 3-Response of GPT-4o; 4-Triplet insertion; 5-Planning output; 6-HTTP response (created triplets/plan).

LLM components related to RRR are deployable on local server, as two language models used as bi-encoder and cross-encoder are not as large as text generation solutions. However, for main part of result generation and prompting, we rely on OpenAI's GPT-4o which is a commercial solution and deployed on OpenAI's cloud infrastructure. The main LLM service unifying RRR with prompting against GPT-4o is also dpeloyed on local server, making use of Langchain library and Ollama for deployment of bi-encoder and cross-encoder models. On the other side, planning-related components and semantic triple store are part of Tasor's semantic planner infrastructure[8] which is part of commercial solution. As a reference execution environment for components deployed on block denoted as local server, we rely on laptop with Intel i5-10300H 2.50GHz CPU, 24GB of RAM and NVIDIA GTX1650 with 4GB of VRAM.

[6] https://www.langchain.com/
[7] https://github.com/ollama/ollama
[8] www.virtuonasoft.com

3.2. Deployment overview

Overview of key functions enabling the semantic annotation of user story relying on LLMs, as well as their arguments and expected outputs are given in Table 2. The deployment of the underlying web services relies makes use of Flask API[9] for Python.

Table 2
Python API overview

Method	Arguments	Output	Description
load_ontology	ontologyPath – Path where the textual file of ontology is stored.	-	Appends the content from given file containg RDF format ontology to the overall text which will be processed by Retrieval and Re-Rank method
search	query – prompt that is used as input for Retrieval and Re-Rank process. It is constructed as combination of pre-defined template and direct input provided by user	Context that will be further used as input to LLM service	Constructs the context by combining the most relevant results (text chunks) returned as outcome of Retrieval and Re-Rank process against the set of planning-relevant ontologies
handle_question	question – user-defined input	Textual response	Relies on search method to get context that will be leveraged for prompt that produces the final LLM-generated answer
__init__	Model – Desired LLM that will be used for response generation (recommended gemma2:9b and GPT-4o)	-	Constructor of the underlying class encapsulating RRR process.

4. Experiments and evaluation

In this section, we aim to evaluate the proposed methodology leveraging LLM for semantic annotation in order to reduce cognitive load required for planning in area of manufacturing. In the first set of experiments, starting from user-provided order description as freeform text, our solution generates semantic knowledge graph segment, which will be further used as input to framework relying on an ontology-driven APS planner in the next step. As reference for semantic annotation, we make use of Tasor Planner ontologies[10] counting 8,503 words combined. Code used in these experiments is available online[11].

The expected value add of the proposed solution is not only to reduce the cognitive load for planning based on ontologies, but also to make the creation of APS inputs faster, as well as to increase overall flexibility of the process, considering that presented LLM-based steps do not depend on the specific ontology for planning. On the other side, in most of traditional approaches, modifications of ontology usually require additional source code changes and updates of the supportive tools, while in our case only the change of input ontology and example knowledge graph are enough.

Creation of several aspects relevant to resources crucial for manufacturing planning process are taken into account:

[9] https://flask.palletsprojects.com/en/stable/api/

[10] Tasor Planner set of ontologies is part of commercial platform. However, results and ideas presented in this paper are not limited by any means because any other set of ontologies can be used instead, in case if some other APS solution is used. The adopted ontology framework is not in the scope of the work presented in this paper.

[11] https://github.com/penenadpi/planrag

- *Employee* – inserting required information about employees involved into production process, such as their name, id within the ERP system, department, team and working position;
- *Order creation* – constructing new manufacturing order, specifying the product ordered, the customer that initiated the order, and deadlines: start and end;
- *Machine* – addition of new machine that can be leveraged within manufacturing process;
- *Activity flow* – defining the multi-step flow of activities which is targeted for planning.

Note that even though manufacturing is selected as the target domain, any other planning and/or optimization problem has the same generic structure.

The following two different variants of ICL prompting approach are evaluated based on previously presented *prompt 1*:

A1) takes user story and ontology excerpt as context into account (context1 in Figure 2), and

A2) in addition to A1 inputs, considers also excerpts from example knowledge graph (practically both context1 and context2 from Figure 2).

Example knowledge graph refers to minimal set of triplets with respect to planning ontologies that is sufficient to describe a full manufacturing order with all of its relevant elements and can be further used as input to APS. Additionally, in the last two experiments, the simpler approach denoted as A1' (based on prompt 2) takes into account only user story and current result, without considering context.

Table 3 summarizes results for the first experiment related to semantic annotation: RDF knowledge graph construction starting from user-provided text. For each of the experiment iterations, we take into account the following aspects: *1) title* of the experiment; *2) text* - input text provided by user; 3) *result* - achieved accuracy for different prompting approaches (denoted as A1 and A2); *4) execution time.* For result accuracy, the ratio of correctly identified RDF elements is taken into account – concepts with their properties and relations. Correctness of generated triplets was evaluated based on human inspection of LLM outputs and their comparison to the triplets from Tasor Planner. Additionally, we also take into account the presence of incorrect, surplus RDF resources (classes, properties and individuals) – which do not exist in any of the production-related ontologies, resulting as outcome of LLM's hallucination effect. Regarding the execution time, two factors are taken into account: retrieval of context (denoted as RAG) and prompt execution (denoted as A1/A2, depending on the approach applied).

Table 3
Experiments and results overview – semantic annotation (Y-yes, N-no, WS-wrong syntax).

Title	Text	Result	A1	A2	Execution time [s]		
Employee creation	*Name of employee is Dusan Kostic. He has id 612. He is member of department project managers. He is member of production team and his position is mechanics designer.*	Classes	1/1	1/1	A1: 6.77	A2: 8.44	RAG: A1: 100 A2: 146
		Properties	1/4	4/4			
		Hallucinated	Y	N			
Order creation	*We have a new order for C1 company and the product we want to produce is P1. The activity starts from 2025-06-01 and ends 2025-07-31.*	Classes	WS	1/1	A1: 8.31	A2: 8.57	RAG: A1: 110 A2: 130
		Properties		5/5			
		Hallucinated		N			
Machine creation	Add new CNC machine with inventory id: CNC_1.	Classes	1/1	-	A1: 6.61	-	RAG: A1: 101 A2:
		Properties	1/1	-			
		Hallucinated	N	-			

							134
Activity flow definition	The production flow contains the following activities: cutting, assembling and packing.	Classes	WS	4/4	A1: 7.72	A2: 7.84	RAG: A1: 108 A2: 131
		Properties		3/3			
		Hallucinated		N			
Employee update	Change the id of employee Dusan Kostic.	Correct update	A1' :Y	A2: Y	A1': 3.1	A2: 6.13	RAG: A2: 101
		Hallucinated	N	N			
Flow extension	Add new activity: preparation before cutting to given flow.	Correct update	A1' : Y	A2: Y	A1': 5.61	A2: 7.88	RAG: A2: 115
		Hallucinated	N	N			

Based on the results from Table 3, it can be noticed that the approach incorporating example graph excerpts within the context (denoted as A2) clearly outperforms the simpler solution leveraging only relevant parts of ontology (denoted as A1). This way, hallucinations are usually avoided considering the fact that correct classes, relation and properties are present within the example graph. Simpler approach A1 is prone to hallucinations, as well as ineffective in more complex scenarios, such as activity flow definition. However, approach A2 with additional context requires also more processing time, taking into account that RRR process will be executed twice, as well as, the length of the prompt itself is also increased this way.

When it comes to graph update scenario, even the simpler approach A1', which does not leverage RAG-extracted context, provides correct result. The simpler approach has advantage considering much faster execution time (no RRR process invocation). The outcome that correct result was produced can be explained by the fact that the update affected the existing elements (like employee's id) which were already present and fully specified in the current result, so only the change of specific values was required.

Table 4 gives insight into examples of the generated results. For each of the covered experiments, the most accurate outcome achieved in the comparison of two approaches is shown.

Table 4
Example triplets created within experiment

Employee creation	@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
	@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
	@prefix kb: <http://example.com/domaineditorfaspasAppKB/> .
	@prefix po: <http://example.com/ProductionOntology/> .
	
	kb:DUSAN-KOSTIC
	    rdf:type po:Employee ;
	    rdfs:label "Dusan Kostic" ;
	    po:hasEmployeePosition kb:MECHANICS-DESIGNER ;
	    po:hasemployeeid "612"^^xsd:integer ;
	    po:memberofdepartment kb:PROJECT-MANAGERS ;
	    po:memberofteam kb:PRODUCTION-TEAM ;
	    po:resourceKey "DUSAN KOSTIC" .
	kb:MECHANICS-DESIGNER
	    rdfs:label "Mechanics Designer" .
	kb:PROJECT-MANAGERS
	    rdfs:label "Project Managers" .
	kb:PRODUCTION-TEAM
	    rdfs:label "Production Team" .

Order	kb:RIDneworder123

creation	rdf:type po:Order ; po:forProduct kb:PRD-P1 ; po:fromCustomer kb:CUST-C1 ; po:hasOrderID "12345"^^xsd:string .
Machine creation	kb:CNC_Machine_1 rdf:type po:RID-1A9557A9-F7E0-4DAA-B0E9-606B1E72F973 ; rdfs:label "CNC Machine 1"@en ; po:hasInventoryId "CNC_1" ; po:resourceKey "HAS_INVENTORY_ID" .

On the other side, in Table 5, the summary of the second set of experiments for ontology-based question answering based on *prompt 3* is shown. Given performance evaluation results are given as average value based on 10 experiment executions. Based on the results, it can be identified that strongest performance is exhibited for attribute retrieval, while answering about relations is a bit more challenging. This phenomenon can be explained that attribute retrieval and direct relation identification are simpler case, considering the locality of information which is extracted from document and injected into context. However, relations in ontology, especially the indirect ones rely on information which is usually not stored inside the document close to each other. Therefore, it can be concluded that the proposed approach exhibits stronger potential when contextual information is expected to be stored within the continuous region inside the document.

Table 5
Experiment results - questions about ontologies.

Experiment	Question	Result	Prompt execution [s]
Attributes list retrieval	Which attributes are relevant for employee definition?	4/4 in 90% cases	1.8
	Which elements are relevant for composite activity?		1.9
Related concepts retrieval	Which concepts are related to order?	3/3 in 80% cases	2.7
Relations retrieval	How employee is related to order?	Correct answer in 70% cases	1.8

5. Discussion

In the existing scientific literature, there are several works that aim to tackle the semantic annotation or knowledge graph construction problem adopting various prompting approaches and different LLMs. However, to the best of our knowledge, there is no other published work focused on production planning case studies that consume semantically annotated data as inputs. On the other side, despite that some of the recent works (such as [17, 18]) mentioned the potential of leveraging synergy with Retrieval Augmented Generation (RAG) for handling larger textual inputs, it is still not so much exploited in the existing solutions.

In [17], an approach to semantic annotation of tabular data was introduced. It examines pure Zero-Shot and Few-Shot prompting, as well as modified methods with entity set provided as part of the context. While promising results were achieved relying on fine-tuned Mixtral 8x7B model, it was stated that the solution struggles when it comes to handling larger tabular inputs. On the other side, the work from [18] leverages combination of LLMs and graph pruning techniques in order to improve performance of situation-specific knowledge graph construction. The approach relies on

GPT-based models (03-mini and 4o), while it is evaluated on two case studies from different domains (ride-hailing and medical science). Despite that the input text are relatively long (thousands of words), the approach is performed iteratively, while it does not tackle the problem of handling large domain describing knowledge graphs as basis of conceptualization within the process of situation-specific knowledge graph construction. Moreover, the work from [19] adopts LLM in synergy with clustering for knowledge graph construction starting from plain text. Despite that it exhibits quite promising results on the benchmark proposed by authors, it is not covering the aspects related to handling larger corpora, but is rather focused on smaller examples.

Some recent work has addressed the problem of adoption of statistical intelligence in integrative solutions for problems that traditionally have been tackled using symbolic intelligence approaches. In [22], an approach similar to one proposed in this paper is used to perform complex manipulation tasks in dynamic environments by combining high-level symbolic plan with low-level motion planning. It proposes an ontology-driven prompt-tuning framework that employs knowledge-based reasoning to refine and expand user prompts with task contextual reasoning and knowledge-based environment state descriptions. The approach proposed in this paper complements that work in the way LLM is used in the system: instead using LLM for plan generation, we use it to generate semantic data inputs into a heuristic-driven planner. In [23], an LLM-based assistant integrated within a modelling platform is proposed to support applications requiring explicit decisions such as in systems engineering tasks. That approach combines similar combination of techniques, such as prompt engineering, few-shot learning, Chain of Thought reasoning, and Retrieval-Augmented Generation to generate accurate and relevant outputs without fine-tuning. However, target application domain is different. Also, semantically annotated data, used as inputs to planning, facilitate seamless integration with data from other sources, such as IoT devices, wearables, ERP, etc. [6].

The symbolic and statistical paradigms of cognition may be considered to be in conflict with each other [24], but we rather consider them as complementing. The approach proposed in this paper addresses one of the possible ways of practical implementations.

Unifying theory of intelligence, including human as well as artificial intelligence, still represents an open challenge [25]. Results presented in this paper give ideas and results that would give some answers necessary to address the challenge.

6. Conclusion

In this paper, we were able to tackle the problem of huge textual data used as basis for LLM prompt context, relying on RRR technique aiming to reduce cognitive load for planning in manufacturing. Despite that LLM-based planning agents are emerging, the lack of human trust is one of main barriers for their practical adoption [20]. Therefore, ontology-based solutions have strong potentially considering the underlying formally grounded approach and representation, making them more trustable [20] than purely LLM-based planning solutions. Therefore, synergy of LLM based approach with knowledge graphs exhibits stronger potential for practical adoption due to transparency (planning steps can be easily reproduced by humans, step by step) and verifiability of planning outcomes. On the other side, pure LLM-based planning solutions usually rely on implicit chain of though and thinking steps that are not directly transparent to humans [21]. However, in future we plan to perform detailed comparison of such planning approaches against the more traditional ones, as well as our semantic-driven solution.

Two groups of experiments were performed: 1) semantic annotation of user story with respect to given domain ontology (including update of the existing graph) 2) question answering about concepts, attributes and relations from given domain ontology. In the first case, two different methods based on in-context learning were compared: 1) ontology collection used in context 2) apart from ontology collection, example RDF graph with respect to those ontologies was used as well. According to the achieved results, it can be noticed that inclusion of relevant excerpts coming from the example knowledge graphs significantly improves the result, leading to both syntax correctness, as well as accurate naming of classes and properties identified. However, considering the additional textual input contained within the context, additional processing time would be needed for both the retrieval and prompt execution. On the other side, the first approach also gives satisfiable results, but in much simpler cases. Regarding the execution time, the first part of the process related to document chunking and retrieval of relevant context information is much longer than the prompting itself. It can be explained by the fact that retrieval happens on local machine with limited resources, while we rely on OpenAI's cloud service for prompt execution. Additionally, in update-related experiments, even the simpler approach with no additional context provided is effective in cases when the target element templates are present within the current result.

When it comes to further extensions of this work, we will aim to explore the strategies and additional steps required that would enable adoption of smaller, locally deployable models, such as focusing LLM-based extraction of raw triplets with respect to simplified ontology representation, while conversion to RDF format would be done using traditional algorithmic approach. Finally, evaluation of the proposed approach for other domains will be performed, as well as more comprehensive evaluation, taking into account additional aspects, such as relationship and value constraints.

Acknowledgements

This work has been supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia [Grant Number: 451-03-137/2025-03/200102]. We thank the anonymous reviewers for their valuable comments and insightful suggestions, which have truly helped improve the quality of this work.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Lin, C. H., et al. The mythical advanced planning systems in complex manufacturing environment. IFAC Proceedings Volumes (IFAC-PapersOnline), 2006, Vol. 39, pp. 703-708.

[2] F. Zanella, C. B. Vaz, Sustainable short-term production planning optimization, SN Comput. Sci. 4 (2023) 824, 1–12. doi:10.1007/s42979-023-02261-7.

[3] S. Liu, H. Cheng, Manufacturing process optimization in the process industry, Int. J. Inf. Technol. Web Eng. (IJITWE) 19 (2024) 1–20. doi:10.4018/IJITWE.338998.

[4] Xu, Yongjun, Xin Liu, Xin Cao, Changping Huang, Enke Liu, Sen Qian, Xingchen Liu et al. "Artificial intelligence: A powerful paradigm for scientific research." The Innovation 2, no. 4 (2021).

[5] P. Haslum, N. Lipovetzky, D. Magazzeni, An Introduction to the Planning Domain Definition Language, Morgan & Claypool Publishers, San Rafael, CA, 2019.

[6] M. Tosic, N. Petrovic, O. Tosic, Wearable Networks for Semantics-Driven FASPAS Approach to Fatigue Management, in: Proc. 7th Int. Balkan Conf. Commun. Netw. (BalkanCom), Ljubljana, Slovenia, 2024, pp. 75–80. doi:10.1109/BalkanCom61808.2024.10557211.

[7] Y. Xu, et al., Artificial intelligence: A powerful paradigm for scientific research, The Innovation 2 (2021) 100179. doi:10.1016/j.xinn.2021.100179.

[8] S. Morales, R. Clarisó, J. Cabot, Impromptu: a framework for model-driven prompt engineering, Softw. Syst. Model. (2025) 1–19.

[9] Q. Dong, et al., A Survey on In-context Learning, in: Proc. 2024 Conf. Empir. Methods Nat. Lang. Process. (EMNLP), Miami, FL, USA, 2024, pp. 1107–1128. doi:10.18653/v1/2024.emnlp-main.64.

[10] V. Gupta, M. Chinnakotla, M. Shrivastava, Retrieve and Re-rank: A Simple and Effective IR Approach to Simple Question Answering over Knowledge Graphs, in: Proc. 1st Workshop Fact Extraction VERification (FEVER), Brussels, Belgium, 2018, pp. 22–27. doi:10.18653/v1/W18-5504.

[11] Zhu, Jiazheng, Xiao Liang, Jiannan Xu, Yingqiang Zhang, Zhonghao Zhang, and Kejia He. "Power Knowledge Question-Answering System Based on Agents, Chain-of-Thought and In-Context Learning." In Proceedings of the 2025 International Conference on Intelligent Systems, Automation and Control, pp. 153-158. 2025.

[12] LangChain, Recursive Text Splitter, LangChain Documentation, 2025. URL: https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/recursive_text_splitter/ (accessed 2025-05-16).

[13] Karpukhin, Vladimir, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. "Dense Passage Retrieval for Open-Domain Question Answering." In EMNLP (1), pp. 6769-6781. 2020.

[14] M. Lu, C. Chen, C. Eickhoff, Cross-Encoder Rediscovers a Semantic Variant of BM25, arXiv preprint arXiv:2502.04645, 2025. URL: https://arxiv.org/abs/2502.04645.

[15] R. Jay, Introduction to LangChain and LLMs, in: Generative AI Apps with LangChain and Python, Apress, Berkeley, CA, 2024. doi:10.1007/979-8-8688-0882-1_.

[16] F. S. Marcondes, A. Gala, R. Magalhães, F. P. de Britto, D. Durães, P. Novais, Using Ollama, in: Natural Language Analytics with Generative Large-Language Models: A Practical Approach with Ollama and Open-Source LLMs, Springer, Cham, 2025, pp. 23–35.

[17] M. Cremaschi, F. D'Adda, A. Maurino, StEELlm: An LLM for Generating Semantic Annotations of Tabular Data, ACM Trans. Intell. Syst. Technol. (Just Accepted) (2025). doi:10.1145/3719206.

[18] S. Alter, Using an LLM to Create Situation-Specific Knowledge Graphs Based on a Domain Knowledge Graph: Practical Possibilities and Semantic Challenges, in: KG4SDSE - Knowledge Graphs for Semantics-Driven Systems Engineering @CAiSE, Vienna, Austria, 2025. URL: https://www.researchgate.net/publication/391704708.

[19] B. Mo, K. Yu, J. Kazdan, P. Mpala, L. Yu, C. Cundy, et al., KGGen: Extracting Knowledge Graphs from Plain Text with Language Models, arXiv preprint arXiv:2502.09956, 2025.

[20] S. Chen, Y. Yang, K. Boggess, S. Heo, L. Feng, U. Topcu, Evaluating Human Trust in LLM-Based Planners: A Preliminary Study, arXiv preprint arXiv:2502.20284, 2025. URL: https://arxiv.org/abs/2502.20284.

[21] K. Wang, J. Li, N. P. Bhatt, Y. Xi, Q. Liu, U. Topcu, Z. Wang, On The Planning Abilities of OpenAI's o1 Models: Feasibility, Optimality, and Generalizability, arXiv preprint arXiv:2409.19924, 2024. URL: https://arxiv.org/abs/2409.19924.

[22] M. U. Din, J. Rosell, W. Akram, I. Zaplana, M. A. Roa, L. Seneviratne, I. Hussain, Ontology-driven Prompt Tuning for LLM-based Task and Motion Planning, arXiv preprint arXiv:2412.07493, 2024.

[23] J.-M. Gauthier, E. Jenn, R. Conejo, Ontology-Driven LLM Assistance for Task-Oriented Systems Engineering, in: Proc. 13th Int. Conf. Model-Based Softw. Syst. Eng. (MODELSWARD 2025), 2025, pp. 383–394. doi:10.5220/0013441100003896.

[24] Y. Maruyama, Symbolic and statistical theories of cognition: towards integrated artificial intelligence, in: Proc. Int. Conf. Softw. Eng. Formal Methods, Springer, Cham, 2020, pp. 129–146.

[25] R. Fjelland, Why general artificial intelligence will not be realized, Humanit. Soc. Sci. Commun. 7 (2020) 1–9.