Towards Generating Complex Alignments with Large Language Models via Prompt Engineering

Guilherme Sousa[1], Rinaldo Lima[2] and Cassia Trojahn[1]

[1]IRIT: Institut de Recherche en Informatique de Toulouse, France
[2]Universidade Federal Rural de Pernambuco, Recife, Brazil

Abstract

Still few ontology matching approaches focus on generating alignments by a Large Language Model (LLM), especially in the complex matching task. This paper proposes an approach that leverages the capabilities of LLMs to perform complex ontology matching. The method integrates subsets of both source and target ontologies into the prompt and, as a response, the LLM generates alignments in the structured EDOAL format, rather than natural language descriptions. This reduction technique, based on the automatic generation of SPARQL queries, tackles the challenge of large prompt sizes, reduces the search space, and enables efficient processing on consumer-grade hardware. This approach is evaluated on the Conference and Geolink datasets from the OAEI complex track, demonstrating improved scalability and the ability to produce well-formed EDOAL. Key contributions include the development of a SPARQL-based prompt engineering strategy and the application of few-shot learning techniques to complex alignment generation.

Keywords

LLM, complex matching, SPARQL

1. Introduction

Ontology matching techniques have predominantly focused on simple alignment tasks that are insufficient to cover real-world needs. As data complexity and their relationships increase, the challenge of accurately aligning schemes and ontologies and generating expressive correspondences becomes more pronounced. Conversely, advancements in the field of Natural Language Processing (NLP) have led to the development of powerful Large Language Models (LLMs), which have demonstrated remarkable capabilities in understanding and generating natural language. Although such models have been explored for simple alignment tasks, there has been a growing interest in their use for complex matching.

A recent work addressing the problem has been proposed in [1]. Their approach harnessed the capacity of LLMs to find alignments between GeoLink Modular Ontology (GMO) and GeoLink Base Ontology (GBO) in the Geolink dataset [2]. In that work, the whole GMO ontology is loaded in the prompt, and the modules in GBO are inserted and asked for the related parts in GMO and the LLM responds in natural language. One of the weaknesses of such approach, however, lies in the natural language output generated by LLMs, which complicates evaluating and verifying the resulting alignment. This requires additional steps to convert from natural language to formats that existing complex evaluation tools can process.

A more general problem, however, is how to automatically modularize an ontology so that each complex entity is defined inside its module. For example, the entity "Paper" may be related to an entity "Author" that probably is not part of the complex "Paper" entity, and filtering out those relations is still a difficult task. In [1], the ontologies are already modularized permitting the search of corresponding modules by the LLM. However, most of the ontologies do not contain that type of modularization

OM-2024: The 19th International Workshop on Ontology Matching collocated with the 23rd International Semantic Web Conference (ISWC 2024), November 11th, Baltimore, USA.

*Corresponding author.
†These authors contributed equally.
guilherme.santos-sousa@irit.fr (G. Sousa); rinaldo.jose@ufrpe.br (R. Lima); cassia.trojahn@irit.fr (C. Trojahn)
0000-0002-2896-2362 (G. Sousa); 0000-0002-1388-4824 (R. Lima); 0000-0003-2840-005X (C. Trojahn)

which limits the generalization of this approach since it will not be able to run in ontologies without predefined modules.

To address these limitations, this paper proposes an approach that selects subsets of both source and target ontologies guided by automatically generated SPARQL queries from PageRank graph nodes. Those subsets are inserted into the prompt and the LLM directly outputs the alignment results in a structured format such as EDOAL rather than natural language descriptions.

By incorporating both ontologies subsets into the prompt, an LLM will have more information to work with since it can combine information from different parts of the ontology to produce each correspondence and produce better alignment results. In addition, the task of delimiting complex entities, combining them, and finding the complex correspondences between entities is passed to the LLM. This approach simplifies the evaluation and verification of the results and provides a more comprehensive and accurate alignment.

A key aspect of our proposal is a technique to reduce the search space by employing the automatic generation of SPARQL queries, as explored in [3][1]. Loading the whole ontologies can lend to very long prompt declarations that common consumer hardware can't execute due to the lack of memory. But relying on these queries, only the relevant information from the ontologies is retrieved, narrowing down the number of entities in the ontologies that the LLM needs to process. This enables the LLMs to generate content using GPUs with lower capacity enhancing the scalability and efficiency of the task while making it feasible to handle large-scale datasets.

The main contributions of this paper are as follows: (i) an approach to complex ontology matching by leveraging the capabilities of LLMs to generate complex alignments in structured format via few-shot prompting and; (ii) a technique based on RAG to reduce the prompt size by creating subsets of the input ontologies guided by SPARQL queries.

The rest of the paper is organized as follows. In Section 2 the research motivations are introduced. In Section 3, the proposed architecture is detailed while explaining all its substeps. In Section 4.1 the experiment settings and results are presented. In Section 5 the related work are discussed. Finally, in Section 6 the conclusion of the paper is presented.

2. Motivating Example

Recent LLM advancements have opened new ways to construct matchers for complex ontology matching tasks. Notably, models like ChatGPT-4o[2] and Claude 3.5[3] have demonstrated the ability to perform complex alignments in controlled, toy problem settings by leveraging the input of entire ontologies into the prompt, as exemplified in Figure 1.

After inserting the ontologies in the prompt, the next step consists of providing the instructions and examples of alignments so the model can learn to write the resulting alignment in the required format as exemplified in Figure 2. The examples provided and the resulting alignment are in EDOAL [4].

The output produced by ChatGPT4o is illustrated in Figure 3. The input ontologies were crafted to contain the concept of *AcceptedPaper* in the source ontology and the concept of *Paper* that have a property *hasAcceptance* leading to a range *Acceptance* in the target ontology. The LLM properly finds similar complex entities between source and target ontologies even without examples of complex correspondences in the prompt. Moreover, the LLM properly combines the entities *Paper*, *hasAcceptance*, and *Acceptance* to produce the complex correspondence while following the proposed file structure in the examples.

This case highlights the potential of LLMs to perform complex ontology matching by directly outputting the alignments in the expected format. However, ChatGPT-4o and Claude 3.5 are private LLMs that have hardware requirements far beyond those available to most consumers. In this sense,

[1]In that work, the notion of Competency Question for Alignment (CQA) has been introduced as a way of expressing user needs in terms of alignments, in the form of SPARQL queries. As it has been also explored in that work, we adopt here an automatic way of generating such queries, referred to as 'SPARQL queries' for short, in this paper.

[2]https://chatgpt.com/

[3]https://claude.ai/new

Given the two ontologies below:

<ontology1> # prefixes : ontology **a owl:Ontology** . :AcceptedPaper **a owl:Class** ; **rdfs:comment** "A paper that has been accepted for presentation at a conference" . :Author **a owl:Class** ; **rdfs:comment** "A person who writes a paper" . :Conference **a owl:Class** ; **rdfs:comment** "An event where papers are presented" . :hasAuthor **a owl:ObjectProperty** ; **rdfs:domain** :AcceptedPaper ; **rdfs:range** :Author . :isPresentedAt **a owl:ObjectProperty** ; **rdfs:domain** :AcceptedPaper ; **rdfs:range** :Conference . **</ontology1>**	**<ontology2>** # prefixes : ontology **a owl:Ontology** . :Paper **a owl:Class** ; **rdfs:comment** "A document submitted to a conference" . :Decision **a owl:Class** ; **rdfs:comment** "A verdict on a paper" . :Acceptance **a owl:Class** ; **rdfs:subClassOf** :Decision ; **rdfs:comment** "A positive decision on a paper" . :hasAcceptance **a owl:ObjectProperty** ; **rdfs:domain** :Paper ; **rdfs:range** :Acceptance . :Author **a owl:Class** ; **rdfs:comment** "A person who writes a paper" . :Conference **a owl:Class** ; **rdfs:comment** "An event where papers are presented" . :hasAuthor **a owl:ObjectProperty** ; **rdfs:domain** :Paper ; **rdfs:range** :Author . :isSubmittedTo **a owl:ObjectProperty** ; **rdfs:domain** :Paper ; **rdfs:range** :Conference . **</ontology2>**

Figure 1: Prompt example containing the source and target input ontologies. They are displayed side by side for easy visualization. The prefixes and labels are omitted for brevity.

enabling open and smaller LLMs to perform this task is a desirable objective. These tests have been corroborated with Llama-3 [4], an open-source LLM available in smaller sizes. It has demonstrated the ability to find correspondences between entities but produced the output in the wrong format even if explicitly instructed to output in EDOAL. By giving examples of alignments, a technique named few-shot learning [5], causes Llama-3 to produce alignments in the right format. These examples highlight the importance of prompt engineering in instructing the LLMs to perform complex matching.

Despite these promising capabilities, significant challenges remain when deploying these models in real-world scenarios. One of the primary obstacles is the substantial size of the prompts due to the whole input of the source and target ontologies in the prompt that can reach the magnitude of millions of tokens. This becomes problematic as consumer-grade GPUs, commonly used in many practical applications, are unable to handle such large inputs efficiently. To deal with these problems, our proposed approach to automatically construct the prompts with examples and in a reduced size is described in the next sections.

3. Proposal

To overcome the challenges of large prompt sizes and limited computational resources, an approach for reducing the search space is proposed. This reduction is based on using generated SPARQL queries to select the most promising parts of an ontology and writing smaller prompts that the LLM can process. Here, the use of automatically generated SPARQL is inspired from [3], where a Competency Question for Alignment (CQA) is a competency question that should (in the best case) be covered by two or more

[4]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

Write a file in EDOAL format containing the complex alignment between the ontology1 and ontology2. You don't need to explain yourself. Just give as response the resulting file without saying anything. Here is one example bellow:

<ontology1>	**<ontology2>**
# prefixes	# prefixes
lib:Book1 **a** lib:Book ;	pub:Book1 **a** pub:Book ;
dcterms:title "The Catcher in the Rye" ;	**dcterms:title** "To Kill a Mockingbird" ;
dcterms:creator lib:Author1 ;	**dcterms:creator** pub:Author1 ;
lib:hasGenre "Fiction" .	**pub:publicationYear** "1960" .
lib:Author1 **a** lib:Author ;	pub:Author1 **a** pub:Author ;
foaf:name "J.D. Salinger" ;	**foaf:name** "Harper Lee" ;
foaf:birthDate "1919-01-01" .	**pub:hasNationality** "American" .
</ontology1>	**</ontology2>**

```
<alignment>
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF #prefixes>
<Alignment>
 <onto1>
  <Ontology rdf:about="http://example.org/library#"/>
 </onto1>
 <onto2>
  <Ontology rdf:about="http://example.org/publishing#"/>
 </onto2>
 <map>
  <Cell>
   <entity1 rdf:resource="http://example.org/library#Book"/>
   <entity2 rdf:resource="http://example.org/publishing#Book"/>
   <relation>=</relation>
   <measure>1.0</measure>
  </Cell>
 </map>
 <map>
  <Cell>
   <entity1 rdf:resource="http://example.org/library#Author"/>
   <entity2 rdf:resource="http://example.org/publishing#Author"/>
   <relation>=</relation>
   <measure>1.0</measure>
  </Cell>
 </map>
</Alignment>
</rdf:RDF>
</alignment>
```

Figure 2: Instruction and few-shot examples added to the prompt to improve quality. The prompt was restructured for better visualization. Prefixes and labels were omitted.

ontologies. However, the user manually writes the CQAs to express their needs, and writing CQAs for each mapping pair is not scalable. In this sense, in this work, the use of automatically generated SPARQL queries is applied to perform the search space reduction.

At the end of the generation step of all prompts, the EDOAL output of the LLM of the same pair of ontologies is later combined. Along with that reduction, a prompt engineering strategy (few-shot) is

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF #prefix>
<Alignment>
          . . .
 <map>
  <Cell>
   <entity1 rdf:resource="http://example.org/ontology1/AcceptedPaper"/>
   <entity2>
    <edoal:Class>
     <edoal:And>
      <edoal:Class rdf:about="http://example.org/ontology2/Paper"/>
      <edoal:PropertyRestriction>
       <edoal:onProperty rdf:resource="http://example.org/ontology2/hasAcceptance"/>
       <edoal:someValuesFrom rdf:resource="http://example.org/ontology2/Acceptance"/>
      </edoal:PropertyRestriction>
     </edoal:And>
    </edoal:Class>
   </entity2>
   <relation>=</relation>
   <measure>1.0</measure>
  </Cell>
 </map>
 <map>
  <Cell>
   <entity1 rdf:resource="http://example.org/ontology1/Author"/>
   <entity2 rdf:resource="http://example.org/ontology2/Author"/>
   <relation>=</relation>
   <measure>1.0</measure>
  </Cell>
 </map>
          . . .
 <map>
  <Cell>
   <entity1 rdf:resource="http://example.org/ontology1/isPresentedAt"/>
   <entity2 rdf:resource="http://example.org/ontology2/isSubmittedTo"/>
   <relation>=</relation>
   <measure>1.0</measure>
  </Cell>
 </map>
</Alignment>
```

Figure 3: Result alignment output by ChatGPT4o. The example of complex correspondence is highlighted. Some correspondences were omitted for brevity.

applied to enable the LLM to perform the complex matching task. The proposed method is detailed in the following sections.

3.1. Reducing Search Space using Automatically Generated Queries

The reduction of space is composed of three main steps, which are detailed in the following.

3.1.1. Identifying Pertinent Entities using PageRank

The first step involves identifying the most relevant entities within the ontologies. In [6] a method to automatically find the most relevant entities and later produce SPARQL queries is proposed and is

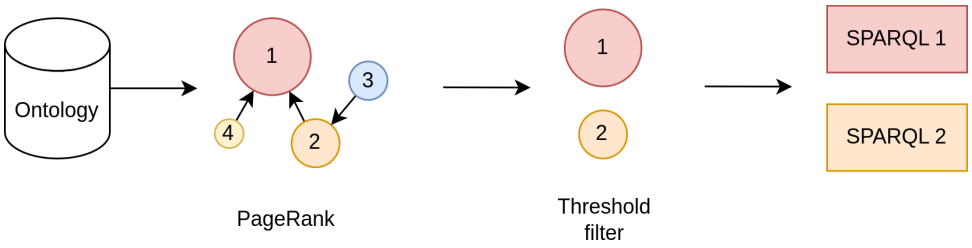

Figure 4: Process of generation of the SPARQL queries by selecting the most relevant nodes in the ontology ranked by PageRank.

based on the number of instances that an entity has. However, not all ontologies contain instances, and some contain few instances of the most relevant classes in the ontology. To improve in this direction, in this work is proposed to apply PageRank as it doesn't need instances to work.

The PageRank [7] algorithm operates on the principle of link analysis, where entities with higher incoming links from other significant entities are considered more relevant. By applying PageRank to the graph structure of the ontologies, a score for each node in the ontology graph is computed and those that have importance over a threshold are selected. These entities are converted to SPARQL queries and will guide the splitting of the ontologies into smaller sizes. This method is applied only in the source ontology to generate the queries that are later used to filter both source and target ontologies.

The PageRank computes the rank of a node by iterating N times all S nodes in the ontology and updating their values in each iteration. The rank of a node is the sum of the ranks of its incoming links divided by their outgoing links. The rank of a link is computed with the equation 1.

$$rank_i(link) = damping * \frac{rank_{i-1}(link)}{outgoing\_links(link)} + \frac{1 - damping}{S} \tag{1}$$

After the final iteration, the rank of all nodes is normalized by dividing the ranks by the maximum rank value, and the nodes with a rank higher than a threshold will be converted to SPARQL queries. The conversion template is based on the work [6] with three cases considered. The classes are injected into the template *SELECT DISTINCT ?x WHERE {?x a <CLASS>.}*, the properties into the template *SELECT DISTINCT ?x ?y WHERE {?x <PROPERTY> ?y.}*, and instances into the template *SELECT DISTINCT ?x WHERE {?x a <INSTANCE>.}*.

3.1.2. Generating Description Documents from Entities

Once the most promising entities are identified, pseudo-documents are generated for each entity in the input ontology. This approach is inspired by the NLFOA matcher [8], which suggests creating textual descriptions of an ontology node's structure to generate embeddings using language models. Evaluated in various studies [9, 10, 11], this type of representation of subgraphs are better utilized by LLMs. While the original method addressed only the representation of simple entities, our work adapts this technique to represent also complex entities. Each pseudo-document includes the target concept along with its parents, children, incoming and outgoing relations, and disjoint concepts.

The template used to produce the documents is given in Figure 5. The template is the same for both simple and complex entities. Complex entities are retrieved from BNodes and processed to generate a text representation. For example, a BNode containing *owl:unionOf* and a chain of *rdf:first* and *rdf:rest* are traversed and the entities are collected in a list and written in a single line of text. For example, the union of *Conference* and *Event* results in the text *UnionOf (Conference, Event)* that is inserted into the field **Name** of the template. The other fields, **Parents**, **Children**, **Incoming Properties**, **Outgoing Properties**, and **Disjoint with** are also lists of entities retrieved from the graph structure of the ontology. The number of entities in each field is limited to produce reduced-size documents.

Target Concept:
Name: {entity_text}.
Parents: {parents}.
Children: {children}.
Incoming Properties: {incoming_properties}.
Outgoing Properties: {outgoing_properties}.
Disjoint with: {disjoints}.

Figure 5: The document template used to represent the ontology entities in textual format.

3.1.3. Creating Sub-Ontology Prompts

The next step is inspired by Retrieval Augmented Generation (RAG) [12] techniques, the final prompts that the LLM will process are generated by combining a subset of both source and target ontologies by expanding the most promising entities retrieved from the ontologies based on the selected generated SPARQL queries. For this step, embeddings of the SPARQL query and the description documents, generated for all entities in each ontology, are computed using the best language model of the MTEB [13] leaderboard [5]. The similarity between the query embedding and the embeddings of the entities documents in the ontology is computed, and the top N most similar entities are selected to be included in the ontology subset. This is a similar process to the retrieval from a vector database [14] where the SPARQL embedding is the query, the embeddings from the documents are the keys and the values are the actual entity that generated the description document. Since the same query guides the search for similar entities in both source and target ontologies, an overlap between the entities is induced by the semantic search of the embedding similarities.

After retrieving the most promising entities, a new ontology is created by inserting the entities subgraphs expanded by traversing the ontology and inserting the same relations triples until a certain depth from the initial entity. This process is applied to source and target ontologies using the same SPARQL query to ensure that entities semantically related to source and target ontologies are present for comparison in the prompt. Finally, a turtle serialization of the subset ontology is produced to be included in the prompt. Serializing the final ontology using turtle format reduces the number of tokens since turtle is less verbose than RDF/XML. The process is illustrated in the Figure 6.

With the sub-ontologies prepared, a prompt is generated by filling in the following template:

```
Given the two ontologies below:
<ontology1>
<ontology2>
Examples of complex alignment between different ontologies:
<ontology1>
<ontology2>
<alignment>
Considering that the input ontologies were filtered to include only the
entities related to the query:
<query>
Write a file in EDOAL format containing the complex alignment between the
ontology1 and ontology2. You don't need to explain yourself. Just give as
response the resulting alignment file without saying anything else.
```

For the experiments, three different variations of the prompt template were tested. The first does

[5] At the time where the experiments were run.

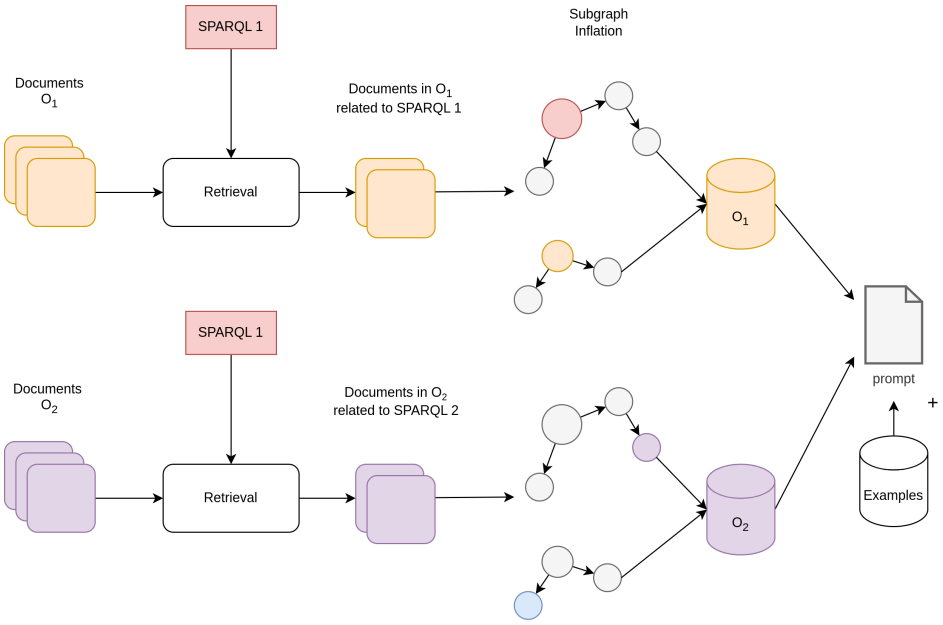

Figure 6: To generate the prompt with the sub ontologies, for each SPARQL query in the source ontology, a retrieval strategy is applied to first: embed the description documents generated and get the most similar ones to the SPARQL query embedding. After this retrieval, the entities are inflated and inserted into a reduced-size ontology that is serialized into the prompt.

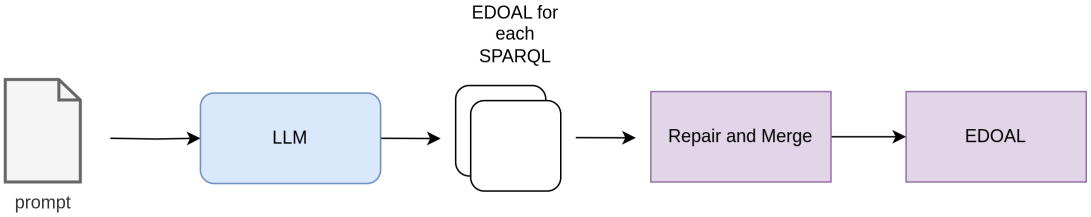

Figure 7: For each prompt, the LLM generates the resulting alignment that is repaired and merged into a single alignment file for each source-target ontology pair.

not include examples, the second includes examples without the SPARQL query used to select the subontology, and the last includes both examples and the query.

3.2. Generation of EDOAL Output

After the generation of the prompts, they are input into the LLM to generate the alignment results in EDOAL. The EDOAL format provides a formal representation of the alignments, capturing complex relationships and correspondences between the entities of the ontologies while facilitating the evaluation of the performance of the alignments produced since the automatic evaluation system used in this work processes that format.

In the generation step, the model is prompted without token sampling, only the most probable token is always selected, with a maximum generation length of 2048 tokens. The generated EDOAL is then repaired in the cases where the model doesn't generate an appropriate file ending (caused by files that are supposed to be larger than 2048 tokens). Then the EDOAL files for all SPARQL queries in the same source and target ontology pairs are combined resulting in a single file. The file contains all the correspondences related to the concepts in all generated SPARQL queries from the same source and target ontology pair. This is the final output of the architecture that will be later evaluated. That process is depicted in Figure 7.

4. Experiments

To evaluate the quality of the alignments produced by the LLM, an experiment was conducted by generating several prompts and evaluating their performance using an automated evaluator [15]. The evaluator assesses the performance of the alignments by comparing the number of similar instances retrieved in each correspondence. Metrics such as precision, recall, and F1-score are used to quantify the performance of the alignment process. In addition, no training or fine-tuning was performed in any model during the experiments, the model's weights are not changed in any way during the execution.

The selected datasets are Conference and Geolink from the OAEI[6] complex track. The selected model for retrieval is Salesforce/SFR-Embedding-2_R used in the process of generating the prompts. The LLM used for the response generation is meta-llama/Meta-Llama-3-8B-Instruct both of them retrieved from HuggingFace [7]. The code used in the experiments can be found in GitLab [8].

4.1. Results and Discussion

The first step in the execution pipeline is the generation of the queries that will narrow the size of the prompts to generate the response by the LLM. The PageRank was run with 10 iterations, a damping factor of 0.8, and a threshold of 0.4 that selects a maximum of 30 entities. The number of entities selected for each ontology is present in Table 1.

Ontology	count
conference/iasted	9
conference/edas	1
conference/Cocus	3
conference/paperdyne	14
conference/crs_dr	5
conference/conference	13
conference/confious	6
conference/MyReview	15
conference/sigkdd	10
conference/PCS	10
conference/OpenConf	30
conference/linklings	3
conference/cmt	9
conference/ekaw	7
conference/MICRO	9
conference/confOf	9
geolink/gmo	5
geolink/gbo	7

Table 1
Number of queries generated through PageRank for each ontology.

After that step, prompt generation was performed as described in Section 3.1.3. It runs with 15 max entities for each field that receives a list of entities in the description document generation. The maximum size of those documents is 4096 characters and the documents are trimmed in case of overflow. The language model used for the embeddings generation is instruction tunned and the instruction passed along with the query was "Given the following SPARQL query, retrieve relevant entities that are related to the query". The entity inclusion considers the top 2 most similar entities to the SPARQL query and inflates them by traversing depth 2 following outgoing triples and depth 1 following incoming triples and adding them into the resulting ontology. A sample of the number of tokens for all prompts

[6]https://oaei.ontologymatching.org/2020/complex/index.html
[7]https://huggingface.co/models
[8]https://gitlab.irit.fr/melodi/ontology-matching/complex/llm-complex-matching

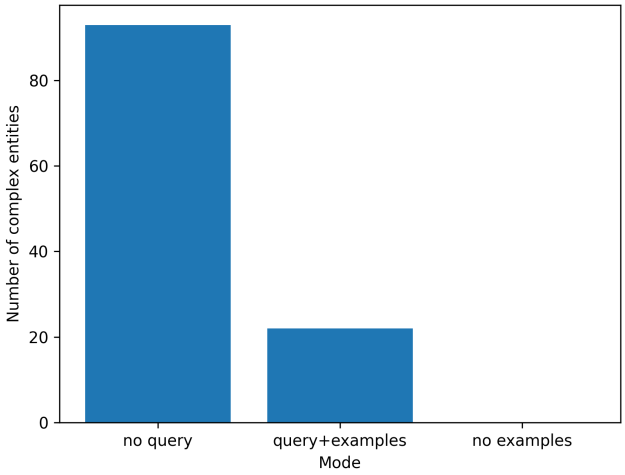

Figure 8: Amount of complex alignments by prompt type.

generated from the pair Cmt-Conference, when tokenized using the Llama3 tokenizer, is present in Table 2. With that strategy, the matching process can run on a 12 GB VRAM GPU.

query	tokens
whole ontologies	11633
c-person	4055
c-preference	3435
c-programcommittee	3926
c-bid	4014
c-decision	3991
c-administrator	3952
c-document	3973
c-paper	4118
c-conference	4790

Table 2
Number of tokens of all prompts generated in the pair Cmt-Conference.

For this experiment, three variations were performed in the LLM prompts. The first variation uses a prompt without examples (LLM, no examples), only containing source and target input ontologies and instructions to generate the final alignment file. The next variation is the usage of two different examples of alignments between sample ontologies (LLM, no query), one containing only simple correspondences and the other containing examples of complex correspondences. The last variation includes examples and instructions explaining what SPARQL query was used to produce the ontologies subsets (LLM, query+examples).

In total as illustrated in Figure 8, 4730 correspondences were produced in the **no examples** variation while no complex correspondences were produced. In the **no query** variation 8567 correspondences were produced while 93 were complex correspondences. And in the **query+examples** variation 2214 correspondences were produced while 22 were complex. The **no query** variation is the one that produces the highest number of correspondences and also the highest number of complex correspondences. The variation **query+examples** generates the lowest number of correspondences however with higher complex entities than the variation **no examples**. The **no examples** variation wasn't able to generate any complex correspondence while a large amount of them are in natural language or in the wrong format. This illustrates the importance of few-shot prompting in the generation of complex correspondences in the right format.

The merged output was evaluated using the automatic evaluator proposed in [15]. It has a CQA-oriented and a precision-oriented evaluation. The CQA oriented requires CQAs to operate while the precision-oriented doesn't require CQAs. The approach was evaluated in the Conference and Geolink datasets. Due to the requirement for CQAs, the Geolink dataset was evaluated only using the precision evaluator and the CQAs for the Conference track are present with the dataset. Moreover, since the evaluator requires instances to work, in the conference dataset, the approach receives as input the Conference dataset and was evaluated in the Populated Conference dataset as it contains instances.

The proposed variations were compared with CANARD [16] matcher and the results of the evaluation are presented in Table 3. The proposed matcher still falls behind the CANARD baseline in most of the metrics. When compared between the variations, the **no query** have higher results in the evaluation mode while the variation **query+examples** achieves higher precision. The variation without examples gets zero results in both CQA-based and precision-based evaluations. As can be seen by these results, providing examples is essential to the model's performance as it learns from the examples how to write the alignments in the EDOAL format.

mode	evaluation	dataset	classical	recall	precision	overlap	f-measure
CANARD	cqa	conference	0.35	0.36	0.47	0.35	0.47
LLM (no examples)	cqa	conference	0.00	0.00	0.00	0.00	0.00
LLM (no query)	cqa	conference	0.15	0.20	0.20	0.23	0.21
LLM (query+examples)	cqa	conference	0.10	0.13	0.12	0.15	0.12
CANARD	prec	conference	0.21	0.26	0.26	0.28	0.26
LLM (no examples)	prec	conference	0.00	0.00	0.00	0.00	0.00
LLM (no query)	prec	conference	0.08	0.09	0.11	0.08	0.82
LLM (query+examples)	prec	conference	0.14	0.14	0.20	0.14	0.78
CANARD	prec	geolink	0.00	0.00	0.00	0.00	0.00
LLM (no examples)	prec	geolink	0.00	0.00	0.00	0.00	0.00
LLM (no query)	prec	geolink	0.00	0.00	0.00	0.00	0.00
LLM (query+examples)	prec	geolink	0.00	0.00	0.00	0.00	0.00

Table 3
Results of the evaluation.

One of the reasons for getting worse results than CANARD is that the proposed matcher generates the wrong entity ID prefixes in some correspondences. That can cause worse results as the evaluator uses the correspondences to retrieve the instances and measures the number of the same instances that the correspondences retrieve, however with the wrong IDs, no instances will be retrieved and that correspondence will be considered as a miss. However, while the proposed matcher still has lower performance than CANARD, it can perform the matching even in the absence of instances, contrary to CANARD which requires the ontologies to be populated to match them.

In the Geolink dataset, both CANARD and the proposed matcher fail to generate correct correspondences. In the case of the proposed matcher, the wrong results are caused by not finding any correct correspondences. Some of the found correspondences, for example, the model generates the correspondence between the property *http://gmo#hasFix* and the image path *http://schema.geolink.org/images/gbo-overview.svg* that is not correct. One possible correspondence found by the proposed matcher is between *http://gmo#InformationObject* and *rdf:about="http://gbo#DigitalObject* but if they fail to share common instances they won't be considered as correct correspondence.

5. Related Work

Ontology (including knowledge graph) matching is a task where natural language information is crucial for development. The rise of LLMs has introduced new methods for enhancing the matching process. While LLMs are being adopted in most pipelines for ontology matching, fewer works deal with directly generating the full alignment output by the LLM for complex matching. This section reviews the works

in the current state of research on utilizing LLMs to generate correspondences for ontology matching. Most existing methods rely on closed-source LLMs presenting reproducibility issues.

OLaLa [17] is a customizable, open-source framework for ontology matching that uses LLM in the matching pipeline. It extracts matching candidates from input ontologies, which are then analyzed by an LLM using a user-defined prompt. The approach combines the LLMs with other matchers to improve the alignments and maintain the requirement of simple alignments (1:1 correspondences). The correspondence candidates are presented to the LLM for binary or multiple-choice decisions with confidence scores aiding in selecting the final correspondences. Various text extractors provide single text outputs for entities, including preferred labels and verbalized RDF triples. While effective in the simple alignment task, the approach wasn't explored for the complex matching case. Also, the decision taken by the LLM considers only entity-level information that can be insufficient to decide the correspondence between entities.

In [1] ChatGPT-4 was used to perform complex matching between the ontologies of GMO and GBO of the Geolink dataset. In that approach, the full GMO ontology was provided as input in the prompt, and similar entities from GBO were matched. In that work, an advantage is taken from the modularization of the ontologies as the complex entity boundaries are already defined in those modules. However, this approach is not accessible in the majority of cases. The first problem is that loading all of the ontologies can consume a considerable amount of memory in GPUs, which most consumers don't have access to. To improve in this direction, divide-and-conquer strategies or compression strategies must be adopted as applied in our proposal. The other issue is that most of the available ontologies are not modularized making the application of that approach without modifications.

More recently, in [18] a pattern-based approach for complex ontology matching has bee proposed. Close to ours, it focuses on the use of SPARQL queries and Large Language Models (LLMs) for detecting and validating complex correspondences between ontologies. Unlike existing methods, this approach avoids lexical matching in the detection phase and relies solely on structural aspects, with LLMs employed only in the final validation step. The approach uses a pattern-based approach to retrieve verbalized candidate correspondences and input into the LLM for verification. However, the output of the LLM is in natural language and the evaluation was performed manually while in this work the LLM output was evaluated automatically.

6. Conclusion

This paper presents an approach to complex ontology matching by leveraging the capabilities of LLMs and optimizing prompt size using a strategy based on automatically generated SPARQL queries. The technique enhances the scalability and efficiency of ontology matching by reducing the number of entities considered, making it feasible to handle large-scale datasets on consumer-grade hardware. Additionally, the proposed method addresses challenges associated with large search spaces in ontology matching, such as modularization and entity combination, through the use of LLMs. For instance, by directly integrating ontologies into the prompt and outputting results in a structured format like EDOAL, the method enables automatic evaluation of alignment results.

Although a qualitative evaluation is still missing, the initial experiments (fully automated) demonstrated that including examples in the prompts is crucial for performing complex matching tasks, as they significantly improve the accuracy and format of alignments generated by LLMs compared to using the same model without examples. Even with the high amount of data used to train those LLMs and possible training with data from OAEI, the experiments with the LLMs conducted in this work demonstrate that the LLMs are not producing the correct EDOAL format without examples in the prompt, indicating that they haven't see enough EDOAL data in the training or are not being able to retain it. While the inclusion of examples notably enhances performance, there remains room for optimization, especially in real-world scenarios where the complexity and scale of ontologies introduce additional challenges. Future work will explore advanced prompt engineering techniques, fine-tuning LLMs specifically for complex ontology matching tasks, and developing models capable of handling

larger prompts without increasing memory consumption.

References

[1] R. Amini, S. S. Norouzi, P. Hitzler, R. Amini, Towards complex ontology alignment using large language models, CoRR abs/2404.10329 (2024). URL: https://doi.org/10.48550/arXiv.2404.10329. doi:10.48550/ARXIV.2404.10329. arXiv:2404.10329.

[2] L. Zhou, M. Cheatham, A. Krisnadhi, P. Hitzler, A complex alignment benchmark: Geolink dataset, in: D. Vrandecic, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee, E. Simperl (Eds.), The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II, volume 11137 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 273–288. URL: https://doi.org/10.1007/978-3-030-00668-6_17. doi:10.1007/978-3-030-00668-6_17.

[3] É. Thiéblin, Do competency questions for alignment help fostering complex correspondences?, in: L. Hollink, F. Osborne (Eds.), Proceedings of the EKAW Doctoral Consortium 2018 co-located with the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France, November 13, 2018, volume 2306 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018. URL: https://ceur-ws.org/Vol-2306/paper8.pdf.

[4] J. David, J. Euzenat, F. Scharffe, C. T. dos Santos, The alignment API 4.0, Semantic Web 2 (2011) 3–10. URL: https://doi.org/10.3233/SW-2011-0028. doi:10.3233/SW-2011-0028.

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[6] É. Thiéblin, Automatic Generation of Complex Ontology Alignments. (Génération automatique d'alignements complexes d'ontologies), Ph.D. thesis, Paul Sabatier University, Toulouse, France, 2019. URL: https://tel.archives-ouvertes.fr/tel-02735724.

[7] M. Bianchini, M. Gori, F. Scarselli, Inside pagerank, ACM Trans. Internet Techn. 5 (2005) 92–128. URL: https://doi.org/10.1145/1052934.1052938. doi:10.1145/1052934.1052938.

[8] F. Schneider, S. Dash, S. Bagchi, N. Mihindukulasooriya, A. M. Gliozzo, NLFOA: natural language focused ontology alignment, in: K. B. Venable, D. Garijo, B. Jalaian (Eds.), Proceedings of the 12th Knowledge Capture Conference 2023, K-CAP 2023, Pensacola, FL, USA, December 5-7, 2023, ACM, 2023, pp. 114–121. URL: https://doi.org/10.1145/3587259.3627560. doi:10.1145/3587259.3627560.

[9] J. Guo, L. Du, H. Liu, Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking, CoRR abs/2305.15066 (2023). URL: https://doi.org/10.48550/arXiv.2305.15066. doi:10.48550/ARXIV.2305.15066. arXiv:2305.15066.

[10] J. Zhao, L. Zhuo, Y. Shen, M. Qu, K. Liu, M. M. Bronstein, Z. Zhu, J. Tang, Graphtext: Graph reasoning in text space, CoRR abs/2310.01089 (2023). URL: https://doi.org/10.48550/arXiv.2310.01089. doi:10.48550/ARXIV.2310.01089. arXiv:2310.01089.

[11] B. Fatemi, J. Halcrow, B. Perozzi, Talk like a graph: Encoding graphs for large language models, CoRR abs/2310.04560 (2023). URL: https://doi.org/10.48550/arXiv.2310.04560. doi:10.48550/ARXIV.2310.04560. arXiv:2310.04560.

[12] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, CoRR abs/2312.10997 (2023). URL: https://doi.org/10.48550/arXiv.2312.10997. doi:10.48550/ARXIV.2312.10997. arXiv:2312.10997.

[13] N. Muennighoff, N. Tazi, L. Magne, N. Reimers, MTEB: massive text embedding benchmark, in: A. Vlachos, I. Augenstein (Eds.), Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023, Association for Computational Linguistics, 2023, pp. 2006–2029. URL: https://doi.org/10.18653/v1/2023.eacl-main.148. doi:10.18653/V1/2023.EACL-MAIN.148.

[14] J. J. Pan, J. Wang, G. Li, Survey of vector database management systems, CoRR abs/2310.14021 (2023). URL: https://doi.org/10.48550/arXiv.2310.14021. doi:10.48550/ARXIV.2310.14021. arXiv:2310.14021.

[15] É. Thiéblin, O. Haemmerlé, C. Trojahn, Automatic evaluation of complex alignments: An instance-based approach, Semantic Web 12 (2021) 767–787. URL: https://doi.org/10.3233/SW-210437. doi:10.3233/SW-210437.

[16] É. Thiéblin, G. Sousa, O. Haemmerlé, C. Trojahn, CANARD: an approach for generating expressive correspondences based on competency questions for alignment, Semantic Web 15 (2024) 897–929. URL: https://doi.org/10.3233/SW-233521. doi:10.3233/SW-233521.

[17] S. Hertling, H. Paulheim, Olala: Ontology matching with large language models, in: K. B. Venable, D. Garijo, B. Jalaian (Eds.), Proceedings of the 12th Knowledge Capture Conference 2023, K-CAP 2023, Pensacola, FL, USA, December 5-7, 2023, ACM, 2023, pp. 131–139. URL: https://doi.org/10.1145/3587259.3627571. doi:10.1145/3587259.3627571.

[18] O. Zamazal, Towards pattern-based complex ontology matching using sparql and llm, in: Proceedings of the 20th International Conference on Semantic Systems (SEMANTiCS 2024), SEMANTiCS, Amsterdam, Netherlands, 2024. Poster paper.