OntoMatch Results for OAEI 2024

Julian Sampels[1]

[1]Technische Universität Berlin, Berlin, Germany

Abstract

This paper presents the results of OntoMatch in the OAEI 2024 competition. OntoMatch is an ontology matching system that combines graph search algorithms with zero-shot prompting of Large Language Models (LLMs) to produce class correspondences. The system follows an iterative approach involving neighbourhood candidate selection, context extraction using graph search techniques, verbalising of context and zero-shot LLM prompting with templates. Each iteration concludes with a cardinality filter to refine the alignments. OntoMatch was evaluated on the OAEI conference benchmark dataset. The results demonstrate the impact of incorporating graph-based contextual information alongside carefully crafted prompt templates, achieving competitive scores and highlighting the effectiveness of LLM-driven approaches for ontology alignment.

Keywords

Ontology Matching, Knowledge Graphs, Prompt Generation, Graph Search, Large Language Model

1. Presentation of the system

1.1. State, purpose, general statement

Ontology matching is a fundamental task in achieving semantic interoperability, aiming to identify correspondences between concepts across heterogeneous ontologies. With the increasing reasoning capabilities of Large Language Models (LLMs) such as Llama2 [1], Mistral [2] and Flan T5 [3], several recent approaches have leveraged LLMs to tackle the ontology alignment problem. Notably, one of the key advantages of using LLMs is the elimination of a task specific training process. Unlike traditional machine learning approaches, which require extensive training on labelled datasets, LLMs can operate effectively by using their pre-trained knowledge. For instance, Norouzi et al. [4] evaluated ChatGPT using various prompt templates, incorporating all ontology triples into prompts to directly identify correspondences. Similarly, OLaLa [5] explored zero-shot, one-shot and few-shot prompting, using templates that include relevant triples asking whether two classes are corresponding.

The proposed ontology matcher, OntoMatch, was first introduced in [6]. It is a new iterative system that combines the benefits of zero-shot LLM prompting with structural graph search algorithms. The following description is taken from this paper.

1.2. Specific techniques used

We propose an ontology alignment pipeline that explores prompt generation leveraging graph search algorithms, as depicted in Figure 1 and detailed in Algorithm 1. The internal graph structure of ontologies enables the representation of elements based on relations within their neighbourhood. We employ iterative neighbourhood candidate selection, followed by a graph search algorithm collecting the contextual neighbourhood information.

At the onset of the process, a pairwise Similarity Computation (see Algorithm 1 Line 1 and Section 1.2.1) is performed for each pair, consisting of one element from each of the first ontology and the second ontology. The High Precision Matcher (see Line 2 and Section 1.2.2) initially aligns tuples that achieve a high similarity score. During each iteration of the matching process, Candidate

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

julian.sampels@campus.tu-berlin.de (J. Sampels)

0009-0007-4021-9591 (J. Sampels)

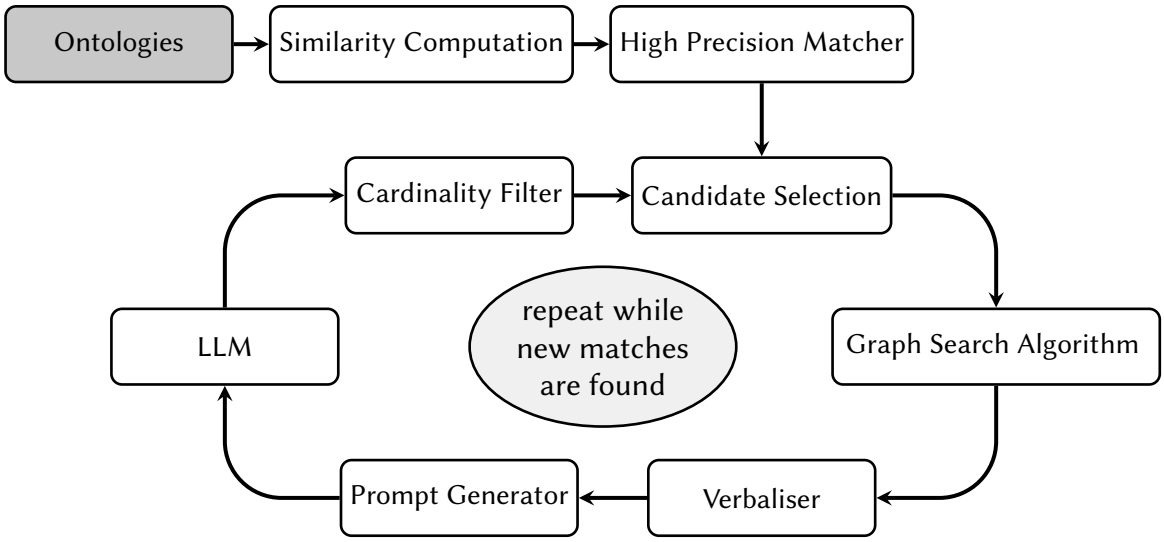

Figure 1: An overview of the framework from [6]

Selection (see Line 6 and Section 1.2.3) is done for evaluation by the LLM. This selection procedure is determined by tuples whose similarity scores meet a minimum threshold and are constrained by the neighbourhood cross products of prior matches. One of the two graph search algorithms (see Lines 7 and 8 and Section 1.2.4) we implemented, namely Random Walk or Tree Traversal, is utilised to extract contextual information from the neighbourhood surrounding each class within both ontologies. This context, represented as triples, requires translation into natural language by the Verbaliser (see Lines 7 and 8 and Section 1.2.7) for evaluation by the LLM. For each candidate pair, a prompt is formulated incorporating the verbalised context of both tuple elements (see Line 9 and Section 1.2.8). These prompts are fed into the LLM (see Line 10 and Section 1.2.9), yielding a *yes* or *no* answer resulting in a mapping. The Cardinality Filter (see Line 14 and Section 1.2.10) reduces the received $(n : m)$-mapping into a $(1 : 1)$-mapping utilising the Hopcroft-Karp algorithm for maximum matchings on bipartite graphs [7]. This loop continues until the LLM evaluates all further discovered candidates as *no match* or no new candidates are found.

Algorithm 1 Matching Pipeline

Require: Source ontology O_1, target ontology O_2, k-hop size k and thresholds t_p, t_c
Ensure: Matching M

1: `similarityComputation`(O_1, O_2) ▷ using Equation (1)
2: $M \leftarrow$ `highPrecisionMatches`(O_1, O_2, t_p)
3: $M_{new} \leftarrow M$
4: **while** $0 <$ `length`(M_{new}) **do**
5: $M_{new} \leftarrow \emptyset$
6: **for** $(c_1, c_2) \in$ `candidates`(M, k, t_c) **do** ▷ using Equation (2)
7: $context_1 \leftarrow$ `verbalise(graphSearch`$(c_1, O_1))$ ▷ using Algorithm 2
8: $context_2 \leftarrow$ `verbalise(graphSearch`$(c_2, O_2))$ ▷ or Random Walk [8]
9: $prompt \leftarrow$ `makePrompt`$(c_1, c_2, context_1, context_2)$
10: **if** `llm`$(prompt) ==$ "*yes*" **then**
11: $M_{new} \leftarrow M_{new} \cup \{(c_1, c_2)\}$
12: **end if**
13: **end for**
14: $M \leftarrow M \cup$ `cardinalityFilter`(M_{new}) ▷ using algorithm from [7]
15: **end while**

1.2.1. Similarity Computation

We apply a cosine similarity score to evaluate each pair within the cross-product of both ontologies, aiming to identify and exclude tuples that are unlikely to match correctly. Before this assessment, the concept names undergo a preprocessing step involving tasks such as resolving camel-cased names and tokenizing class names split by '_' or '-'. Following the preprocessing step, the similarity computation with Equation (1) is conducted on their vector embeddings computed by the Sentence BERT (SBERT) model, utilising the "all-MiniLM-L6-v2"[1] transformer variant of SBERT.

$$S(c_1, c_2) \coloneqq \frac{\Omega(c_1)}{\|\Omega(c_1)\|_2} \cdot \frac{\Omega(c_2)}{\|\Omega(c_2)\|_2} \quad \forall c_1, c_2 \in O_1 \times O_2, \tag{1}$$

where $O_1 \times O_2$ is the cross product of both Ontology classes and $\Omega(\cdot)$ the vectorised embedding of the concept names.

1.2.2. High Precision Matcher

A class tuple $(c_1, c_2) \in O_1 \times O_2$ is considered a high precision match if their similarity score, as defined in Equation (1), exceeds the threshold of 0.95, denoted as $0.95 \leq S(c_1, c_2)$. The reason why we chose a 0.95 similarity score as threshold in the High Precision Matcher instead of 1.0 is that most concepts may comprise spelling mistakes and be written in either US or UK English. These high precision matches are considered as initially new matches M_{new} in the first iteration of the procedure (see Algorithm 1 Line 3).

1.2.3. Candidate Selection

Matching exclusively based on the results from the LLM is computationally intensive, as it requires running a prompt for each pair of classes. To maintain efficiency, we preselect candidates based on the similarity score S defined in Equation (1) and the neighbourhood of the previously matched classes denoted by M. Analogous to the High Precision Matcher, the similarity score must reach $\mathit{threshold} \leq S(c_1, c_2)$. In our candidate selection process, we consider *threshold* as 0.4 without any specific computation among test cases, since SBERT embeddings consider the direct meaning of words due to its attention mechanism. Furthermore, in order to qualify as a candidate, a tuple (c_1, c_2) must satisfy the two additional conditions: firstly, the tuple entries c_1 and c_2 must each be unmatched; secondly, there must exist a tuple $(c_1', c_2') \in M$ that is already matched such that c_1 is in the k-hop reachable neighbourhood $N_k(c_1')$ and analogously c_2 in the neighbourhood $N_k(c_2')$. This entire candidate selection process is precisely formulated in the following equation,

$$C_k(M) \coloneqq \{(c_1, c_2) \in \left(N_k(c_1') \times N_k(c_2')\right) \setminus M_\times \mid (c_1', c_2') \in M \wedge \mathit{threshold} \leq S(c_1, c_2)\}, \tag{2}$$

where the exclusion of $M_\times \coloneqq \{(c_a, c_b) \mid (c_a, \cdot) \in M \wedge (\cdot, c_b) \in M\}$ leads to candidate sets consisting solely of unmatched classes. The k-hop reachable neighbours $N_k(v)$ of v comprise all the neighbours of v within a distance of at most k from v for $k \in \mathbb{N}$.

1.2.4. Graph Search Algorithm

Extracting context from the ontologies requires traversing the neighbourhood of a class in the corresponding Knowledge Graphs (KG). We leverage two simple and well-known algorithms: (i) a Random Walk algorithm akin to the approach proposed by Gosselin et al. [9] and (ii) a Tree Traversal algorithm designed to extract a partial spanning tree from the KG within fixed boundaries. The reason why we chose these two algorithms is that previous works [8, 10] have utilised them in their neighbouring collection parts. In both algorithms, we consider the neighbouring classes connected by a *rdfs:subClassOf* (isParentOf for general to specific direction), *rdfs:subClassOf* (isChildOf for specific to general direction) or *owl:equivalentClass* (isEquivalentTo) relation to the node. Additionally, property relations

[1]The model is available at https://www.sbert.net/docs/pretrained_models.html.

connected with *rdfs:domain* and *rdfs:range* are included. Previous works [8, 11, 12] also take into account these relations to compute representations (or embeddings) of class names in their ontology matching approaches.

1.2.5. Random Walk Algorithm

A random walk is a sequence of nodes where each next node is selected randomly from the unvisited neighbours of the preceding node. The Random Walk algorithm aims to generate $b \in \mathbb{N}$ random walks of length $l \in \mathbb{N}$, all starting from the same root concept. Excluding the root node, the random walks are pairwise disjoint. The random walks are stored as a list of triples containing the previous concept, the next concept and their relation. We adapted the Random Walk algorithm used by Gosselin et al. [8].

1.2.6. Tree Traversal Algorithm

The objective of the Tree Traversal algorithm is to construct a partial spanning tree rooted in the node from which the context is extracted. To optimise efficiency and ensure contextual relevance, the tree is limited in breadth and depth. Our Tree Traversal algorithm, as described in Algorithm 2, is based on a breadth-first search (BFS) approach. The breadth limitation is achieved by reducing the outgoing branches for each node. Similarly, the depth constraint is enforced by a maximal height parameter that controls the distance of each node from the root node in the tree. In situations where the number of neighbours exceeds the breadth limitation, a random selection approach is used.

Algorithm 2 Tree Traversal Algorithm

Require: Ontology, root concept c_r, maximal branches b and height h_{max}
Ensure: Tree triples T

$Visited \leftarrow \emptyset$ ▷ set of visited nodes
$T \leftarrow \emptyset$ ▷ Tree with triples $(concept1, relation, concept2)$
$Queue \leftarrow [(0, c_r)]$ ▷ Queue containing tuples $(height, concept)$
while $0 <$ `length(`$Queue$`)` **do**
  $(h_v, v) \leftarrow$ `dequeue(`$Queue$`)`
  $n \leftarrow$ `length(neighboursOf(`v`))` $\setminus Visited$
  for $i \coloneqq 1$ **to** `minimum(`b, n`)` **do**
    $w \leftarrow$ `randomItem(neighboursOf(`v`))`
    $T \leftarrow T \cup \{(v, \text{relation of } v \text{ to } w, w)\}$
    if $w \notin Visited \wedge h_v < h_{max}$ **then**
      $Queue \leftarrow Queue + [(h_v + 1, w)]$
      $Visited \leftarrow Visited \cup \{w\}$
    end if
  end for
end while

1.2.7. Verbaliser

In order to transform the triples of the classes to be aligned into easily understandable natural language, we utilise the *Graph2Text* model developed by Amaral et al. [13]. This innovative approach operates by taking a series of triples as its input and generates coherent, human-readable sentences (see Table 1) that encapsulate the information conveyed by these triples. The inclusion of this step is not strictly necessary, as LLMs are capable of comprehending the triples in their original format.

1.2.8. Prompt Generator

The specific prompt for the selected candidates is formulated by filling in the information in one of the four prompt templates, as described in Items 1 to 4. `concept1` and `concept2` are placeholders

Table 1
Examples for verbalised triples

Ontology#Concept	Verbalised triple
cmt#PaperFullVersion	PaperFullVersion is the parent of both Paper and Meta-Review. Paper is also the parent of Document which is the child of Meta-Review and PaperFullVersion.
edas#MealEvent	A meal event is a parent of a conference dinner and a child of an academic event, which is parent of a social event that, in turn, is parent of an excursion.

representing the names of the classes, while `context1` and `context2` are derived from graph search algorithms applied to these ontologies, either with or without a Verbaliser.

1. **Comprehensive task description with question and contextual information:**
   ```
   In this task, we are given two concepts along with their definitions from
   two ontologies. Our objective is to provide ontology mapping for the provided
   ontologies based on their semantic similarities.
   ontology1#concept1: context1, ontology2#concept2: context2
   Does the concept concept1 correspond to the concept concept2? yes or no:
   ```
2. **Short task description with question and contextual information:**
   ```
   Classify if two concepts refer to the same real world entity.
   This is the context for the first concept concept1: context1
   This is the context for the second concept concept2: context2
   Do these concepts concept1 and concept2 refer to the same real world entity?
   yes or no:
   ```
3. **Short task description with contextual information but without question:**
   ```
   Classify if the following two concepts are the same.
   First concept concept1: context1
   Second concept concept2: context2
   Answer yes or no:
   ```
4. **Basic prompt for reference without providing contextual information:**
   ```
   Is concept1 and concept2 the same? The answer which can be yes or no is:
   ```

1.2.9. Large Language Model (LLM)

The prompts generated in the previous stage of this pipeline might be sent to encoder-decoder LLMs, such as Flan T5 [14] or decoder only models, e.g., Mistral [2] and Llama2 [1]. This enables us to determine whether two concepts from a pair in an ontology correspond or not. The LLMs respond with a *yes* if the two concepts to be aligned exhibit semantically similar contexts; otherwise, they indicate *no* (See Table 2). The chosen LLM, Flan T5-XL, has achieved remarkable results in Knowledge Graph Construction tasks, such as domain-specific ontology construction from text [15] and relation extraction between entities in a sentence [16]; therefore, we leverage this model in our approach.

1.2.10. Cardinality Filter

In our approach, we utilise the well-known Hopcroft-Karp algorithm for maximum matchings on bipartite graphs [7] to efficiently generate a $(1 : 1)$ mapping from our $(n : m)$ mapping provided by the LLM. This algorithm ensures a one-to-one correspondence with a worst-case complexity of $\mathcal{O}\left(|V|^{5/2}\right)$.

1.3. Adaptations made for the evaluation

The final configuration involved setting a similarity threshold of 0.95 for the High Precision Matcher and 0.4 for Candidate Selection. The k-hop size was fixed at 2, while a depth of 3 and a breadth of 2 was selected for the Tree Traversal algorithm, constrained by the Verbaliser's length limitations.

Table 2
An example for our generated prompts on the conference tracks and the Flan T5-XL LLM response

Case	Prompt	LLM
cmt#PaperFull-Version;confer-ence#Abstract	In this task, we are given two concepts along with their definitions from two ontologies. Our objective is to provide ontology mapping for the provided ontologies based on their semantic similarities. cmt#PaperFullVersion: PaperFullVersion is the parent of both Paper and Meta-Review. Paper is also the parent of Document which is the child of Meta-Review and PaperFullVersion. conference#Abstract: Extended abstract is the parent of Extended abstract. Does the concept "PaperFull-Version" correspond to the concept "Abstract"? yes or no:	yes

1.4. Link to the system

OntoMatch is available under the GNU General Public License (GPL), Version 3 [17]. The source code can be accessed from the GitHub repository: https://github.com/JulianSampels/OntoMatch.

2. Results

This section discusses the results[2] of OntoMatch for the OAEI 2024 tracks. OntoMatch shares some similar LLM concepts with OLaLa [18] and, for similar reasons, is not designed for multilingual input. One specific OntoMatch factor is the used verbaliser (see Section 1.2.7), which is only capable of processing English. Furthermore, as the proposed system is relatively new, we have some compatibility issues with other OAEI tracks. Consequently, it is only capable of handling OAEI's conference tracks at this state of development.

2.1. Conference

The conference track consists of seven ontologies with reference alignment cases, all focused on the domain of conference organisation. The calculated average reference alignment density is approximately 2.2×10^{-3}, which serves as an important metric for graph-structure based neighbourhood candidate approaches, where new candidates are typically found near to previous matched classes.

This track includes several reference alignment sets: M1 focuses solely on classes, M2 targets properties and M3 covers both classes and properties. Since OntoMatch currently matches classes exclusively, we present the results based on the reference alignment variant *rar2-M1*[3], which contains only violation-free classes.

OntoMatch achieved an overall F1 score of 0.63, which is above the StringEquiv baselines and equivalent to edna. It demonstrated a high precision score of 0.82, while maintaining a good recall of 0.51 in the OAEI 2024 Campaign.

3. General comments

3.1. Comments on the results

OntoMatch is designed and programmed with the idea of modularity, providing the advantage of uncomplicated component replacement, adaptation and interchangeability. This modular architecture facilitates the potential for addressing the language barriers, enabling for substitution of specific LLM components within the framework with alternative parts that better align with the desired functionalities.

[2]The results for the OAEI 2024 conference tracks are available at https://oaei.ontologymatching.org/2024/results/conference/.
[3]The results for the *rar2-M1* reference alignments of OAEI 2024 conference tracks are available at https://oaei.ontologymatching.org/2024/results/conference/eval.html#rar2-M1.

When comparing with previous evaluations reported in [6], the results show notable deviations, which could be attributed to differences in evaluation settings. Contributing factors to this discrepancy could include variations in the LLM prompting process across systems, as well as potential issues related to the verbalisation component. Both elements require compatibility with the underlying hardware system to function correctly and avoid errors.

3.2. Discussions on the way to improve the proposed system

In future work, we plan to enhance compatibility with additional OAEI tracks and explore the use of other LLMs, such as Llama2 [1] and Mistral [2]. Furthermore, we aim to optimise the parameters for the graph search algorithms and the similarity threshold during the execution of the High Precision Matcher. Specialised settings and components for speed could extend the framework's capabilities and performance.

References

[1] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., Llama 2: Open foundation and fine-tuned chat models, arXiv preprint (2023). doi:10.48550/arXiv.2307.09288.

[2] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al., Mistral 7b, arXiv preprint (2023). doi:10.48550/arXiv.2310.06825.

[3] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al., Scaling instruction-finetuned language models, Journal of Machine Learning Research 25 (2024) 1–53. URL: https://www.jmlr.org/papers/volume25/23-0870/23-0870.pdf.

[4] S. S. Norouzi, M. S. Mahdavinejad, P. Hitzler, Conversational ontology alignment with ChatGPT, CEUR-WS (2023). doi:10.48550/arXiv.2308.09217.

[5] S. Hertling, H. Paulheim, OLaLa: Ontology matching with large language models, in: Proceedings of the 12th Knowledge Capture Conference 2023, K-CAP '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 131–139. doi:10.1145/3587259.3627571.

[6] J. Sampels, S. Efeoglu, S. Schimmler, Exploring prompt generation utilizing graph search algorithms for ontology matching, in: Knowledge Graphs in the Age of Language Models and Neuro-Symbolic AI, IOS Press, 2024, pp. 2–19. doi:10.3233/SSW240003.

[7] J. E. Hopcroft, R. M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM Journal on computing 2 (1973) 225–231. doi:10.1137/0202019.

[8] F. Gosselin, A. Zouaq, Sorbet: A siamese network for ontology embeddings using a distance-based regression loss and BERT, in: International Semantic Web Conference, Springer, 2023, pp. 561–578. doi:10.1007/978-3-031-47240-4_30.

[9] F. Gosselin, A. Zouaq, Sebmatcher results for OAEI 2022, CEUR-WS (2022). URL: https://ceur-ws.org/Vol-3324/oaei22_paper12.pdf.

[10] Y. He, J. Chen, H. Dong, E. Jiménez-Ruiz, A. Hadian, I. Horrocks, Machine learning-friendly biomedical datasets for equivalence and subsumption ontology matching, in: International Semantic Web Conference, Springer, 2022, pp. 575–591. doi:10.5281/zenodo.6510086.

[11] V. Iyer, A. Agarwal, H. Kumar, VeeAlign: A supervised deep learning approach to ontology alignment, CEUR-WS (2020). URL: https://ceur-ws.org/Vol-2788/oaei20_paper13.pdf.

[12] S. Efeoglu, GraphMatcher: A graph representation learning approach for ontology matching, CEUR-WS (2022). doi:10.48550/arXiv.2404.14450.

[13] G. Amaral, O. Rodrigues, E. Simperl, Prove: A pipeline for automated provenance verification of knowledge graphs against textual sources, Semantic Web (2022) 1–34. doi:10.3233/SW-233467.

[14] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, IEEE Transactions on Knowledge and Data Engineering (2024). doi:10.1109/TKDE.2024.3352100.

[15] N. Mihindukulasooriya, S. Tiwari, C. F. Enguix, K. Lata, Text2KGBench: A benchmark for ontology-driven knowledge graph generation from text, in: International Semantic Web Conference, Springer, 2023, pp. 247–265. doi:10.1007/978-3-031-47243-5_14.
[16] S. Efeoglu, A. Paschke, Retrieval-augmented generation-based relation extraction, 2024. doi:10.48550/arXiv.2404.13397.
[17] Free Software Foundation, GNU general public license, version 3, https://www.gnu.org/licenses/gpl-3.0.html, 2007.
[18] S. Hertling, H. Paulheim, OLaLa results for OAEI 2023, CEUR-WS (2023). URL: https://ceur-ws.org/Vol-3591/oaei23_paper7.pdf.