# Enriching Ontologies with Disjointness Axioms using Large Language Models*

Elias Crum[1], Antonio De Santis[2], Manon Ovide[3], Jiaxin Pan[4], Alessia Pisu[5], Nicolas Lazzari[6], and Sebastian Rudolph[7]

[1] Ghent University, Belgium elias.crum@ugent.be
[2] Politecnico di Milano, Italy antonio.desantis@polimi.it
[3] University of Tours, France manon.ovide@univ-tours.fr
[4] University of Stuttgart, Germany jiaxin.pan@uni-stuttgart.de
[5] University of Cagliari, Italy alessia.pisu96@unica.it
[6] University of Bologna, Italy nicolas.lazzari2@studio.unibo.it
[7] TU Dresden, Germany sebastian.rudolph@tu-dresden.de

**Abstract.** Ontologies often lack explicit disjointness declarations between the classes, which are critical for sophisticated reasoning and consistency-checking in knowledge graphs. In this study, we explore the potential of Large Language Models (LLMs) to enrich ontologies by identifying and asserting class disjointness axioms. Our approach leverages the implicit knowledge embedded in LLMs, using prompt engineering to assess their capability in classifying ontological disjointness. We validate our methodology on the DBpedia ontology, examining both open and closed-source LLMs. Our findings suggest that LLMs, when guided by effective prompt strategies, can reliably identify disjoint class relationships, thus streamlining the process of ontology completion without extensive manual input. For comprehensive disjointness enrichment, we propose a process that takes logical relationships between disjointness and subclass statements into account to reduce the number of calls to the LLM. This work provides a foundation for future applications of LLMs in automated ontology enhancement and offers insights into optimizing LLM performance through strategic prompt design.

**Keywords:** Large Language Models · Disjointness Learning · Ontology Enrichment

## 1 Knowledge Graphs and Generative AI

Manual curation of Knowledge Graphs (KGs) is a tedious and error-prone endeavor requiring comprehensive domain expertise. Thus, it seems tempting to use Large Language Models (LLMs) as "ground truth oracles" in KG creation, replacing human experts. Yet, it is widely known that LLMs are not always accurate; they may fabricate false and/or incoherent information. Errors introduced into a KG through such inaccuracies are most detrimental if they are

* Technical report of the Slytherin team from ISWS 2024 led by Sebastian Rudolph.

located in the terminological part of the KG, for instance, the class hierarchy. Open questions regarding how, on one hand, to improve the reliability of LLM answers through prompt engineering, and how, on the other hand, formal reasoning methods can be applied to safeguard and optimize the knowledge acquisition process against LLM inaccuracies, by detecting contradictions.

## 2 Introduction

When reasoning over a knowledge graph, ontologically dictated class relationships, specifically disjointness statements, can be leveraged for sophisticated reasoning and consistency-checking tasks. Despite their usefulness, these relationships are regularly not explicitly reflected within an ontology.

**Definition 1 (Class Disjointness, Disjointness in OWL).** *Two classes* C *and* D *are* disjoint *iff* $C^{\Im} \cap D^{\Im} = \emptyset$*, where* $C^{\Im}$ *and* $D^{\Im}$ *are the semantic interpretations of the classes* C *and* D*, respectively. Disjointness in the web ontology language OWL and OWL 2 is stated by the axiom* $\texttt{DisjointClasses}(C, D)$.

From an ontological perspective, two classes are considered "disjoint" if and only if their taxonomic overlap is empty. For example, "Cat" and "Car" are disjoint classes because, semantically, an entity that is a cat cannot be a car and vice versa. Such a relationship is defined as *ontological disjointness*. Notably, the lack of an observed occurrence of an entity belonging to two different classes may not classify them as disjoint. For instance, although historically, no woman has served as US President, a woman may be elected as the US President in the future. Therefore, "Woman" and "US President" are not ontologically disjoint classes; instead, we might label them as *materially disjoint* due to the absence of material evidence demonstrating their non-disjoint nature.

Because ontological class descriptions are recorded as natural language, we aim to assess the potential of LLMs to make these class disjointness classifications while assessing the impact of prompt engineering on classification validity.

Introducing disjointness axioms significantly aids in consistency checking and the automatic evaluation of entity class membership within a knowledge base.

Despite the importance of disjointness assertion among classes, research on 1,275 ontologies shows that only 97 of them include disjointness assertions [18]. Two obstacles to adding disjointness assertions are the expected assumption of self-evident logical relationships between classes as well as the time-consuming task of manually labeling disjoint classes. Therefore, semi-automated labeling of disjoint classes could be advantageous. Recent papers [17, 16, 11] propose supervised and unsupervised models using various features in disjointness axioms. However, the generalizability of these methods is limited to their specific datasets and cannot be implemented on a large scale. Additionally, the sophisticated feature engineering required hinders their practical application. Therefore, a method that functions independently of feature design and dataset restrictions is highly important.

We hypothesize that through the use of prompt engineering, LLMs will be able to classify ontologically disjoint classes with high validity in both positive (two classes are ontologically disjoint), and negative (two classes are not ontologically disjoint), cases. We tested our hypothesis on the DBpedia ontology[8] using both closed and open-source LLMs. We propose a method that intertwines the LLM-based disjointness classification with basic logical inferencing to increase efficiency, maintain consistency, and minimize the number of calls to the LLM.

## 3 Research Questions

This article is dedicated to answering the following research questions:

RQ1: Can LLMs help enrich ontologies with class disjointness axioms?
RQ2: Which LLM prompts work better for disjointness discovery?
RQ3: How can we take logical relationships into account to reduce the interactions with the LLM?

## 4 Related Work

*Disjointness Learning.* Disjointness Learning Models can be divided into supervised and unsupervised approaches. Among the unsupervised settings, [12] follows the *strong disjointness assumption* [3]: the children of a common parent in the subsumption hierarchy should be considered disjoint. They introduced a pinpointing algorithm to identify minimal sets of axioms that need to be revised to make an ontology coherent, thereby enriching appropriate disjointness statements. However, it neglects background knowledge, which could be beneficial in identifying disjoint classes. [11] proposes an unsupervised approach based on concept learning and inductive classification. It employs a hierarchical conceptual clustering method capable of providing intensional cluster descriptions and utilizes a novel form of semi-distances over individuals in an ontological knowledge base, incorporating available background knowledge.

Among the supervised methods, [17, 16] gather syntactic and semantic evidence, such as association rules, negative association rules, and correlation coefficients, from various sources to establish a strong foundation for learning disjointness. However, their works exploit background knowledge and reasoning to a limited extent. Subsequent work, DL-Learner [7], uses Inductive Logic Programming (ILP) for learning class descriptions, including disjointness.

Still, as of yet, disjointness acquisition with LLMs seems much underexplored.

*Large Language Models.* Recently, large language models (LLMs) have shown strong capabilities in NLP and beyond. LLMs rely on pre-training Transformer [15] models over large-scale corpora. Pre-trained context-aware word representations achieve state-of-the-art performance on various downstream tasks and

[8] https://DBpedia.org/ontology/

set the "pre-training and fine-tuning" learning paradigm. Training corpora for early LLMs such as BERT [5] were relatively small, and fine-tuning was often necessary for specific downstream tasks.

Further research unveiled that scaling up the size of pre-trained language models and datasets often yields significant performance gains on downstream tasks and solves a range of complex problems: GPT-3 [1] performs comparably to fine-tuned models on numerous tasks through few-shot learning. Without any parameter updates, it can answer questions via in-context learning, using examples provided by users. GPT-3.5, comes with enhanced capabilities and incorporates reinforcement learning from human feedback (RLHF). GPT-4 [8], extends beyond text input to include multimodal signals. Meta AI introduced the collection of LLaMA models [13, 14] with four sizes. Other LLMs like Claude, Gemini [10], and Mixtral [4] have also garnered significant attention.

*Prompt Engineering.* Designing effective prompts for large language models (LLMs) has become a crucial element of maximizing LLM effectiveness. Well-described prompt engineering strategies include zero-shot [9], few-shot [1], and chain of thought [2] prompting. *Zero-shot prompting* [9] provides task descriptions to the LLMs without providing input-output solution examples. It leverages the LLMs' pre-existing knowledge to generate answers. *Few-shot prompting* [1] supplements the task description with several input-output examples to guide the LLMs' generation. High-quality examples in few-shot prompting have been shown to yield significant improvements on complex reasoning tasks. *Chain-of-Thought (CoT) prompting* [19] facilitates coherent and step-by-step reasoning processes. CoT prompting involves decomposing a complex question into a series of simpler logical reasoning questions. This method mimics how humans break down complicated problems and reason through them.

## 5 Resources

*Ontology.* For our analysis, we required a reference ontology for class disjointness classification. Ideally, the reference ontology should have some disjoint classes in its description, preferably a specific disjoint class property such as `owl:disjointWith`, and a moderately sized set of classes to ensure relationship diversity for experimentation. These criteria were used to maximize the generalizability of this exploratory work and encourage its use for guiding future studies. We avoided highly domain-specific ontologies, ontologies with class labels that require expert knowledge to interpret, and ontologies without labeling properties, such as `skos:prefLabel` or `rdfs:label`.

Ultimately, we chose DBpedia's ontology[9] because of its general popularity and conformity with dataset minimal requirements.

*Large Language Models.* For our study we employed several LLMs, which will be described in more detail in Section 7.1.

[9] `https://DBpedia.org/ontology/`, often referred to with the `dbo:` namespace, which we will omit hereafter

## 6 Proposed approach

Our approach relies on several logical correspondences:

**Proposition 1.** *Let $\mathcal{K}$ be a knowledge base and let $C_1$, $C_2$, $D_1$, $D_2$ be classes of $\mathcal{K}$ that such that the following statements follow from $\mathcal{K}$: (i) $C_1$ and $C_2$ are disjoint, (ii) $C_1$ is a subclass of $D_1$, (iii) $C_2$ is a subclass of $D_2$. Then $K$ also entails that $D_1$ and $D_2$ are disjoint.*

We exploit this property to use subclass relationships from $\mathcal{K}$ to deduce class disjointness statements from existing class disjointness statements. This way we avoid posing redundant disjointness queries to the underlying LLM.

**Proposition 2.** *Let $\mathcal{K}$ be a knowledge base and let $C_1$, $C_2$, $C$ be classes of $\mathcal{K}$ that such that the following statements follow from $\mathcal{K}$: (i) $C_1$ and $C_2$ are disjoint, (ii) $C$ is a subclass of $C_1$, (iii) $C$ is a subclass of $C_2$. Then $\mathcal{K}$ also entails that the class $C$ is empty.*

We exploited this property indirectly under the assumption that any named class $C$ in the considered ontology is supposed to have instances – which seems to be a reasonable assumption since, otherwise, the definition of the class appears to be pointless. In that case, any two classes that have a common subclass must be not disjoint.

**Proposition 3.** *Let $K$ be a knowledge base, let $C_1$, $C_2$ be classes and let $e$ be an individual of $\mathcal{K}$ that such that the following statements follow from $\mathcal{K}$: (i) $C_1$ and $C_2$ are disjoint, (ii) $e$ is an instance of $C_1$, (iii) $e$ is an instance of $C_2$. Then $\mathcal{K}$ is unsatisfiable.*

Again, this property can be exploited by noting that any two classes having common instances must not be disjoint. All these considerations and efficiency considerations lead to the method we suggest, detailed in the following:

**Expected Input**: a knowledge base $\mathcal{K}$ containing a class hierarchy with natural language labels for its classes. The knowledge base might or might not contain defined disjointness axioms and/or more complex axiomatizations.
**Expected Output**: a set $\mathcal{D}$ of class disjointness axioms, such that all valid disjointness statements logically follow from $\mathcal{K} \cup \mathcal{D}$ and no invalid disjointness statements follow from it.

1. Create a list $\mathcal{L}$ of all pairs of classes $(C_1, C_2)$ such that $C_1$ is lexicographically smaller than $C_2$ *(exploit the symmetry of disjointness)*. In $\mathcal{L}$, the entries are to be labeled "disjoint" or "not disjoint". Initially, all are labeled "unknown".
2. For every disjointness statement from $K$ between classes $D_1$ and $D_2$, declare every pair $(C_1, C_2)$ from $\mathcal{L}$ disjoint for which one of the two is the case:
   (i) $C_1$ is a subclass of $D_1$ and $C_2$ is a subclass of $D_2$.
   (ii) $C_1$ is a subclass of $D_2$ and $C_2$ is a subclass of $D_1$.
3. For every pair $(C_1, C_2)$ of classes in $\mathcal{L}$ still declared "unknown", check if $C_1$ and $C_2$ have joint subclasses. If so, declare the pair $(C_1, C_2)$ "not disjoint".
   (NB: This includes e.g. the case where $C_2$ is a subclass of $C_1$ since then $C2$ is the joint subclass.)

4. For every pair $(C_1, C_2)$ of classes in $\mathcal{L}$ still declared "unknown", query $K$ for joint instances of $C_1$ and $C_2$. If they exist, declare $(C_1, C_2)$ "not disjoint".
5. Repeat the following as long as there are "unknown" pairs left in $\mathcal{L}$:
   (a) Pick one pair $(D_1, D_2)$ from $\mathcal{L}$ declared "unknown"
   (there may be different picking strategies to try here – we leave this for future work)
   (b) **Ask the LLM if $D_1$ is disjoint from $D_2$ using a reliable prompt**
   (c) If agreed, declare every $(C_1, C_2)$ from $\mathcal{L}$ "disjoint" for which:
   (i) $C_1$ is a subclass of $D_1$ and $C_2$ is a subclass of $D_2$ or
   (ii) $C_1$ is a subclass of $D_2$ and $C_2$ is a subclass of $D_1$.
   (d) If denied, declare every $(C_1, C_2)$ from $\mathcal{L}$ "not disjoint" for which:
   (i) $D_1$ is a subclass of $C_1$ and $D_2$ is a subclass of $C_2$ or
   (ii) $D_1$ is a subclass of $C_2$ and $D_2$ is a subclass of $C_1$.
6. Let $\mathcal{D}^*$ consist of axioms `DisjointClasses`($C_1$,$C_2$) for all $(C_1, C_2)$ from $\mathcal{L}$.
7. Determine the minimal subset $\mathcal{D}$ of $\mathcal{D}^*$ such that $\mathcal{K} \cup \mathcal{D}$ entails $\mathcal{D}^*$.

## 7 Results and Discussion: Preliminary Experiments

### 7.1 Experimental setting

We performed our experiments on both Closed-Source LLMs and Publicly Available LLMs. For the former, we chose GPT-3.5 and GPT-4o as they are the most recent and powerful LLMs. For the latter, we chose Llama3 8B and Llama3 70B from Meta. We used the GPT models from the available websites. GPT-3.5 is the free version, and GPT-4o is under academic license. For the Llama models, we performed our experiments on a server equipped with RTX4090 and used ollama as the LLM provider. The models were quantized to 8bit.

Inside DBpedia's ontology, we identified manually 4 pairs of ontological disjoint classes using the `owl:disjointWith` property: (`Event`, `Person`), (`Place`, `Agent`), (`TimePeriod`, `Person`), (`SpatialThing`, `Work`). These classes will be our ground truth to test our LLMs.

We also considered some human-obvious (non)disjointness statements that are not explicitly described in DBpedia to see if LLMs could understand and give an appropriate answer. We established two short lists of pairs of DBpedia classes: a list of positive examples, where we expect the LLM to say that the classes are disjoint, and a list of negative examples, where we expect the LLM to deny that the classes are disjoint.

| positive examples | negative examples |
|---|---|
| (`Place`, `Person`) | (`BasketballPlayer`, `BaseballPlayer`) |
| (`Continent`, `Sea`) | (`Garden`, `HistoricPlace`) |
| (`Gene`, `Movie`) | (`President`, `BeautyQueen`) |
| (`Planet`, `Star`) | (`MeanOfTransportation`, `Reptile`) |
| (`BaseballLeague`, `BowlingLeague`) | (`Castle`, `Prison`) |

*Prompting.* Overall, we adopted four prompting strategies in the prompting design process: task description, zero-shot prompting, one-shot prompting, and chain-of-thoughts prompting.

**Table 1.** Performance on disjointness detection for LLMs and prompt strategies.

| Model | Prompt | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| *GPT-3.5* | Zero-shot Naive | 0.60 | 0.60 | 0.60 | 0.60 |
| | Zero-shot Task Description | 0.60 | 0.56 | **1.00** | 0.71 |
| | One-shot Task Description | 0.50 | 0.50 | **1.00** | 0.67 |
| | CoT Task Description | 0.50 | 0.50 | **1.00** | 0.67 |
| *GPT-4o* | Zero-shot Naive | 0.80 | 0.80 | 0.80 | 0.80 |
| | Zero-shot Task Description | **0.90** | **0.83** | **1.00** | **0.91** |
| | One-shot Task Description | 0.60 | 0.56 | **1.00** | 0.71 |
| | CoT Task Description | 0.80 | 0.80 | 0.80 | 0.80 |
| *Llama3-8B* | Zero-shot Naive | 0.70 | 0.67 | 0.80 | 0.73 |
| | Zero-shot Task Description | 0.50 | 0.50 | 0.80 | 0.62 |
| | One-shot Task Description | 0.80 | 0.71 | **1.00** | 0.83 |
| | CoT Task Description | 0.60 | 0.57 | 0.80 | 0.67 |
| *Llama3-70B* | Zero-shot Naive | 0.80 | 0.71 | **1.00** | 0.83 |
| | Zero-shot Task Description | 0.50 | 0.50 | 0.40 | 0.44 |
| | One-shot Task Description | 0.80 | 0.80 | 0.80 | 0.80 |
| | CoT Task Description | - | - | - | - |

- **Task Description** This strategy involves detailing the disjointness detection task and providing LLMs with comprehensive background information.
- **Zero-shot Prompting** This strategy directly asks LLMs to answer a question without any input-output examples, relying entirely on the LLMs' pre-existing knowledge.
- **One-shot Prompting** We use a single input-output example in this strategy, allowing the example to serve as a gold standard for the expected output of the LLMs.
- **Chain-of-Thoughts (CoT) Prompting** We incorporate a suggestion, such as "Let's think step by step" or "Reason step-by-step," into the original prompt, as proposed by [6].

In total, we designed four kinds of prompts: zero-shot naive prompts without any strategies, zero-shot task description prompts, one-shot task description prompts, and Chain-of-Thought task description prompts. The templates for these prompts can be found in the Appendix A.

### 7.2 Results

We used accuracy, precision, recall, and F1-Score to evaluate LLMs' performance in Disjointness Detection.[10] We did not implement CoT prompts on Llama3-70B because of the high response time. The overall result is shown in Table 1. Given the results, we can state the following observations:

[10] We are aware that it is methodologically questionable to use these statistical measures on small data sets. We still find them indicative in a qualitative sense.

1. In general, GPT-4o with the zero-shot Task Description prompting strategy performs best across all evaluation metrics. This validates the powerful reasoning ability of GPT-4o for the disjointness detection task.
2. The Task Description Strategy seems to improve the performance of closed-source LLMs (GPT models) while decreasing the performance of publicly available LLMs (Llama models). We conjecture that the training corpus of GPT models contains more material related to disjointness, enabling GPT to extract useful information more effectively with the provided descriptions.
3. Notably, the CoT strategy does not perform as well as we anticipated, despite its success in improving reasoning ability on various complex tasks. We believe this is due to the rather straightforward nature of the disjointness detection task, which typically requires only a simple comparison between two classes.

## 8 Conclusion and Future Work

Our study shows that LLMs can roughly identify and assert disjointness axioms in ontologies, with GPT-4o being the top-performing model. By harnessing their inherent background knowledge and employing strategic prompt engineering, we showed that these models could classify ontological disjointness with minimal manual intervention. This capability simplifies ontology management and supports more robust reasoning in knowledge graphs. Our findings underscore the potential of LLMs as valuable tools for the automated enrichment of ontologies, which encourages future exploration and innovation in this domain.

As immediate future work, the approach proposed in Section 6 would have to be evaluated as a whole on some appropriate ontologies, including DBpedia. These studies would also consider more LLMs, use a larger dataset, and would be followed by an evaluation of the obtained results by human domain experts.

As further future work, we could improve and expand our strategies for testing disjointness. It could be worthwhile to look into heuristics for – given a large list of disjointness candidate pairs – picking those entries that are particularly "promising". One option would be to follow the strong disjointness assumption [3] and pick "sibling classes", that is classes $A$ and $B$ that have a common direct superclass $C$. Furthermore, it could be interesting to test classes that have just one or two examples of non-disjointness, as these instances may be errors to be removed from the KG. On another note, one could develop strategies of gauging the reliability of an LLM response by rephrasing the question asked, e.g. by exploiting that disjointness is symmetric, by pairing the prompt *(1) Is it true that no A is a(n) B?* with the prompt *(2) Is it true that no B is a(n) A?*, to see if the LLM responses are coherent or not. Our prompts can also be improved and refined. We could add a role description in prompts to see if it makes answers better. We also thought of using elements like "at the same time" in prompts to check the temporality of the disjointness or "theoretically" to force abstraction.

## References


[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pages 1877–1901.

[2] Banghao Chen, Zhaofeng Zhang, Nicolas LangrenÃ©, and Shengxin Zhu. *Unleashing the potential of prompt engineering: a comprehensive review*. 2024. arXiv: `2310.14735 [cs.CL]`.

[3] Ronald Cornet and Ameen Abu-Hanna. "Usability of expressive description logics–a case study in UMLS." In: *Proceedings of the AMIA symposium*. American Medical Informatics Association. 2002, page 180.

[4] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, LÃ©lio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, ThÃ©ophile Gervet, Thibaut Lavril, Thomas Wang, TimothÃ©e Lacroix, and William El Sayed. *Mixtral of Experts*. 2024. arXiv: `2401.04088 [cs.LG]`.

[5] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of NAACL-HLT*. 2019, pages 4171–4186.

[6] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. *Large Language Models are Zero-Shot Reasoners*. 2023. arXiv: `2205.11916 [cs.CL]`.

[7] Jens Lehmann. "DL-Learner: learning concepts in description logics". In: *The Journal of Machine Learning Research* 10 (2009), pages 2639–2642.

[8] R OpenAI. "Gpt-4 technical report. arxiv 2303.08774". In: *View in Article* 2.5 (2023).

[9] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, Jeffrey Dean, and Sanjay Ghemawat. "Language Models are Unsupervised Multitask Learners". In: *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150.

[10] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context". In: *arXiv preprint arXiv:2403.05530* (2024).

[11] Giuseppe Rizzo, Claudia D'Amato, and Nicola Fanizzi. "An unsupervised approach to disjointness learning based on terminological cluster trees". In: *Semantic Web* 12.3 (2021), pages 423–447.

[12] Stefan Schlobach. "Debugging and semantic clarification by pinpointing". In: *European Semantic Web Conference*. Springer. 2005, pages 226–240.

[13] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[14] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[16] Johanna Völker, Daniel Fleischhacker, and Heiner Stuckenschmidt. "Automatic acquisition of class disjointness". In: *Journal of Web Semantics* 35 (2015), pages 124–139.

[17] Johanna Völker, Denny Vrandečić, York Sure, and Andreas Hotho. "Learning disjointness". In: *The Semantic Web: Research and Applications: 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007. Proceedings 4*. Springer. 2007, pages 175–189.

[18] Taowei David Wang. "Gauging Ontologies and Schemas by Numbers." In: *EON@ WWW*. 2006.

[19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2022. arXiv: `2201.11903 [cs.CL]`.

# A Prompt Templates

**Table 2.** Templates for Different Prompting Strategies.

| **Prompting Strategy** | **Templates** |
|---|---|
| Zero-shot Naive | Can a(n) A be a B? |
| Zero-shot Task Description | This is a *question about ontological disjointness*, answer only with "yes" or "no". Are the classes A and B ontologically disjoint? |
| One-shot Task Description | *Example of ontological disjointness is Q*: Are the classes "Cat" and "Car" ontologically disjoint? A: Yes.<br>Please answer the following question only with "yes" or "no". Q: Are the classes A and B ontologically disjoint? |
| CoT Task Description | This is a question about ontological disjointness, final answer can only be "yes" or "no". Are the classes A and B ontologically disjoint? Also, *include your reasoning step-by-step.* |