# ReWiSe: Relation-Wise Self-consistency for LLM Probing

Edouard Albert-Roulhac[1,2], Amal Zouaq[1,2]

[1] LAMA-WeST Lab
[2] Polytechnique Montréal

**Abstract**

Large Language Models (LLMs) learn facts and general knowledge from unstructured data. Evaluating the knowledge within an LLM and extracting it from its parametric memory are challenging tasks as models hallucinate and use a stochastic generation process. The LM-KBC competition challenges participants to propose an approach to construct a Knowledge Graph with an LLM and no external corpus. In this work, we propose a new approach called ReWiSe which uses chain-of-thought reasoning and relation-wise self-consistency. We create a synthetic chain-of-thought dataset with reasoning paths designed for the limited set of relations in the challenge. This synthetic dataset is then used as few-shot samples to make predictions. Chain-of-Thought reasoning provides gains for relations such as `countryLandBordersCountry`, where structured strategies (e.g., geographic enumeration) guide the model toward more complete answers. We finally propose to adapt self-consistency with a relation-wise approach that adapts to relation cardinality and schema. Our results show that relation-wise self-consistency leads to strong performance gains on the LM-KBC benchmark. Using 20 sampled generations, ReWiSe won the 2025 edition of LM-KBC with a Macro-F1 score of 44% (the baseline is 21%). The implementation is available at https://github.com/Lama-West/ReWiSe.

## 1. Introduction

Knowledge graphs (KGs) are fundamental knowledge representations for tasks such as question answering and semantic search. A Knowledge Graph (KG) is defined as a set of triples $(h, r, t)$ where $h$ is the head entity, $r$ is the relation and $t$ is the tail entity. They are typically constructed manually via crowdsourcing which is difficult to scale or through text mining which requires a high-quality text database. World knowledge graphs are incomplete in practice since adding all triples of the world's knowledge is not feasible. This is why automatic methods for KG construction is an active research field.

Extracting knowledge from Large Language Models (LLMs) and storing it in a knowledge base is a promising path to exploit LLM knowledge. This knowledge is learned during pretraining on large unstructured text corpora. Once knowledge is organized, for example in a Knowledge Graph, it can be easily retrieved and used in downstream applications. This knowledge base construction task is challenging as LLM knowledge is stored in a diffuse way across their parameters and thus hard to access directly. Querying LLMs with prompts to extract this knowledge is a simple solution but it is often insufficient because of the following challenges. First, text generation is a non-deterministic and prompt-dependent process, and a model may output inconsistent responses to semantically identical inputs [1, 2, 3]. Second, LLMs are prone to hallucinations, confidently asserting wrong facts [4]. Finally, determining what facts the model does or does not have in its parametric memory is an open problem [5].

These challenges motivate the LM-KBC competition, which evaluates LLMs on their ability to complete a knowledge graph using only parametric knowledge.

**The LM-KBC challenge.** The objective of the Language Models - Knowledge Base Construction (LM-KBC) challenge [6] is to use a LLM to complete triples in a Wikidata-based KG. Wikidata is a free

*Joint proceedings of KBC-LM and LM-KBC @ ISWC 2025*

*Corresponding author.

†These authors contributed equally.

edouard.albert-roulhac@polymtl.ca (E. Albert-Roulhac); amal.zouaq@polymtl.ca (A. Zouaq)

online multilingual collaborative KG. It is general-purpose and contains over 115 billion triples [7]. Given a head entity and a relation $(h, r)$, the task is to use an LLM to output the list of all tail entities $t_i$ such that $(h, r, t_i)$ is in the KG. Triples can have 0, 1 or many tail entities.

In the 2025 edition, model fine-tuning and using external corpora with RAG are disallowed. The Qwen3-8B model is imposed.
Evaluation is done with Macro-F1 score. A $5\%$ margin is allowed for numerical entities. String matching is used for non-numerical entities.

**Relations.** The dataset for the 2025 edition is a subset of Wikidata. It features 6 relations. Each one only appears 10 to 100 times in both the training and the validation splits of the dataset. The relations are as follows:

- `countryLandBordersCountry` (68 input pairs per split): This list can have 0, 1 or multiple elements.
- `personHasCityOfDeath` (100 input pairs per split): Can be empty.
- `hasCapacity` (100 input pairs per split): Numeric (integer) answer.
- `hasArea` (100 input pairs per split): Numeric (real number) answer.
- `awardWonBy` (10 input pairs per split): The list can be long (up to 224).
- `companyTradesAtStockExchange` (100 input pairs per split): Can be empty.

This setting presents several challenges. First, LLM outputs are stochastic and sensitive to prompt phrasing, making it hard to reliably extract structured knowledge from a single query. Second, prior work shows that reasoning via intermediate steps can improve factual accuracy, but constructing such reasoning paths at scale remains non-trivial. Finally, relations in a knowledge graph follow a schema—ranging from single entities to long lists or numerical values—which complicates aggregation across generations. In this work, we explore the following **research questions**:

1. **Can we teach a systematic way to retrieve parametric knowledge through Chain-of-Thought?**
   We build HumanCoT which contains examples of systematic Chain-of-Thought reasoning for the model to reproduce the same kind of reasoning during inference.
2. **How can synthetic data generation be leveraged in data-scarce settings for LLM probing?**
   We generate synthetic reasoning paths that lead to the correct answer for every example of the training set and use them in independent predictions with the self-consistency process.
3. **How can we adapt self-consistency to relation schema in a Knowledge Graph, especially when there is no assumption on the cardinality of the relation?**
   We aggregate the outputs of several model calls at the entity level. When there is no assumption on the cardinality of the relation, we introduce a threshold which represents the confidence of the model in predicting each entity. This enables to tune the aggregation strategy relation-wise.

## 2. Related Work

Several methods that tackle KG completion rely on accessing external knowledge at inference time [8] or models trained on specific knowledge with the same entities and relations as those used at inference time [9]. These methods and similar ones listed in [10] are not allowed in this edition of the challenge which is about extracting the knowledge from the parametric memory of the LLM itself, not using the LLM to process information retrieved in an existing database.

**Prompt-based methods** attempt to use natural language to query the model, associating prompts to relations following the approach of [11]. These prompts can be templates for masked language models or questions for auto-regressive models. The search for an optimal prompt is a key challenge in this

context. AutoPrompt [12] optimizes text prompts for this purpose. The optimization of prompts can also be done with soft tokens that are trained through gradient descent to retrieve information as in Prefix Tuning [13]. This is not allowed in the challenge as it is a fine-tuning approach. Some work has also been done to find optimal few-shot examples to provide to the models [14].
A motivation for the search of a good prompt is the fact that similar prompts may not produce similar answers [1]. This is still an open research problem.

**Chain-of-Thought reasoning** enables the model to perform intermediate reasoning steps before producing a final answer, effectively trading computational cost for improved answer quality [15]. The intermediate context generated by the model functions as a form of working memory. Recent models—particularly Qwen3 [16]—are explicitly trained to respond using Chain-of-Thought reasoning. This approach is valuable as it allows the model to break down complex problems into steps and to externalize contextual knowledge stored in its parametric memory via this working memory [17], which can aid in arriving at a more accurate final answer. However, generating reasoning paths at scale is an open problem. We address this by generating a synthetic dataset tailored to each relation.

**Ensembling and consistency methods** leverage multiple prompts and generation attempts from large language models (LLMs) to improve robustness to prompt variability. For example, [18] generate paraphrased versions of an initial prompt to diversify inputs. Self-evaluation techniques allow the model to assess and either confirm or refute its previous outputs [19].

Similarly, Self-Consistency [20] builds on the inherent stochasticity of text generation. It produces multiple reasoning paths and answers to a given question, then selects the most consistent answer using majority voting. This approach assumes a single correct answer—typically a discrete choice in a multiple-choice question or a numerical value. Universal Self-Consistency [21] extends this idea by proposing a more general framework capable of handling a broader range of answer types. Instead of relying on majority voting, it prompts an LLM to select the most consistent answer from among the generated candidates, allowing plain-text answers to be considered valid.

However, KG completion introduces a more structured setting than general QA, which must be taken into account. Each relation in a KG follows a specific schema. Some relations expect numerical outputs, while others require lists of object entities—sometimes with cardinality constraints. Exploiting this underlying structure is essential for effective answer aggregation in this context, which is what we propose in this work. To the best of our knowledge, self-consistency has not previously been applied to structured prediction tasks like KG completion.

**Contributions.** In order to leverage the full training set and the reasoning potential of the LLM, we propose to use a synthetic data generation process, in which we automatically generate reasoning paths leading to a correct answer. This synthetic dataset is then used to teach the model how to complete triples based on the relation involved. In addition, we propose a relation-wise version of self-consistency. It exploits the stochasticity of generative models and adapts to relations' schema.

# 3. Methodology

## 3.1. Baseline

The baseline presented by the competition uses few-shot prompting. Each relation is associated to a question and 5 examples of questions and answers from the training set are provided before asking a question from the test set. The model generates a list of answers directly. The baseline's performance is shown in Table 1. It reaches a Macro-F1 score of 20.0%.

**Question Prompting**, using a question which expects the tail entities as answer.

```
System (instructions)
Given a question, your task is to provide the list of answers
without any other context. If there are multiple answers,
separate them with a comma. If there are no answers, type
"None".
User (example 1)
In which city did Jan Turski die?
Assistant
Warsaw
User (example 2)
In which city did Basina Kloos die?
Assistant
None
...
User (question)
In which city did Jeroen Brouwers die?
Assistant
```

| Relation | Macro-p | Macro-r | Macro-F1 |
|---|---|---|---|
| awardWonBy | 0.173 | 0.027 | 0.043 |
| companyTradesAtStockExchange | 0.230 | 0.561 | 0.215 |
| countryLandBordersCountry | 0.653 | 0.873 | 0.628 |
| hasArea | 0.180 | 0.180 | 0.180 |
| hasCapacity | 0.030 | 0.030 | 0.030 |
| personHasCityOfDeath | 0.100 | 0.550 | 0.100 |
| **All Relations** | 0.209 | 0.401 | **0.200** |

**Table 1**
The official competition baseline gets a Macro-F1 score of 20.0%

### 3.2. Relation-Wise Self-consistency

We propose a new approach called Relation-Wise Self-consistency (ReWiSe) that combines synthetic reasoning paths with entity-level aggregation of predictions. ReWiSe proceeds through four main steps summarized below.

**HumanCoT.** We manually craft reasoning paths, called HumanCoT, for a few triples per relation. Each reasoning path is free-form text which guides the model toward producing the correct answer. The goal is that the model copies the reasoning strategies from HumanCoT in the inference process.

**SyntheticCoT.** Using HumanCoT as few-shot examples, we generate synthetic reasoning paths and answers for other training triples. Only outputs leading to correct answers are kept.

**Inference Generation.** For each test triple, we generate multiple reasoning paths and corresponding answers. This ensembling, known as Self-Consistency, addresses stochasticity in both model generation and few-shot example selection.

**Relation-Wise Aggregation.** Finally, we discard the reasoning paths and aggregate answers at the entity level. Depending on the schema of the relation, different strategies are tested on the validation set and the best one is kept for each relation.

These steps are detailed in the following sections.

## 3.3. HumanCoT dataset

In order to impose a systematic way to reason about each relation, we need a dataset with correct reasoning paths associated to each triple. Instead of writing them for every triple of the training set - which is not scalable - we write only a few examples per relation in a HumanCoT dataset. In total, the HumanCoT dataset contains 20 reasoning examples across the six relations. These examples are used in few-shot prompting settings to create a SyntheticCoT dataset. The dataset is available in our repository[1].

The reasoning path starts with a <think> tag and ends with a </think> tag allowing for easy parsing of the generated text. We will denote $CoT$ the reasoning path and $a$ the answer.

Here is, for each relation, the systematic reasoning that was used to build HumanCoT dataset. By using examples from HumanCoT to build SyntheticCoT, and then using SyntheticCoT examples for inference, we aim to propagate the way of reasoning about relations consistently throughout the pipeline.

We rely on heuristics and empirical tests to build the reasoning paths. It should start with a short definition of the subject entity and general knowledge about it. The model should identify critical knowledge to answer the question [22]. Triples included in HumanCoT are chosen to represent the schema of each relation. For numeric relations, HumanCoT triples represent the range of possible outputs. If a relation cannot have a correct answer, a HumanCoT triple will have no correct answer.

Overall, two groups of relations emerge: those that could be addressed through reasoning steps to guide the model, and those that relied on factual knowledge and could not necessarily be resolved through reasoning alone.

The first group includes `countryLandBordersCountry`, `personHasCityOfDeath` and `awardWonBy`.

**countryLandBordersCountry.** The strategy adopted here is to reason geographically by checking land neighbors around the clock. For this relation, we provide 4 examples of human reasoning paths.

**Example**: Which countries share a land border with Kenya?

**Chain-of-Thought** <think> Kenya is a country located in East Africa, and its eastern side is bordered by the Indian Ocean. Therefore, Kenya has no land borders on its eastern side. To determine its land borders, we need to consider neighboring countries in the region. First, looking to the north, Ethiopia is directly adjacent to Kenya, making it a clear land neighbor. Moving northeast, Somalia lies next to Kenya and extends along the eastern Horn of Africa. To the west, Kenya shares a border with Uganda. While the Democratic Republic of the Congo is nearby, it does not border Kenya directly because Uganda lies between them. To the northwest, Kenya shares a small border with South Sudan but it does not touch Sudan, near the Ilemi Triangle region. Finally, to the south, Kenya is bordered by Tanzania, with the border extending along the Serengeti-Maasai Mara area. </think> Ethiopia, Somalia, South Sudan, Tanzania, Uganda.

**personHasCityOfDeath.** For this relation, the strategy is first to determine whether the person has died, then to retrieve the city of death. A frequent LLM error is to output a city where the person lived rather than died. Importantly, death is definitive: if the model has seen evidence of a person's death, it can remember it, whereas mentions of someone being alive don't guarantee they're still living.

For this relation, we provide 4 examples of human reasoning paths.

**Example**: In which city did Ricky Nelson die?

**Chain-of-Thought** <think> Ricky Nelson was an American musician and actor. He is known for his country rock style. He lived in Los Angeles but died in an aircraft crash near Dallas, Texas. </think> Dallas

[1] https://github.com/Lama-West/ReWiSe/blob/main/data/dataset2025/cot/human_cot_single.csv

**awardWonBy.** This relation is challenging as the number of ground truth outputs can be large. Empirically, we found that generating the whole list of outputs is too hard. Instead, our strategy is to generate coherent subgroups of award winners, for example by grouping them by year of award, gender, or achievement. The model has to identify a subgroup and generate recipients' names. When the model generates multiple subgroups, they can later be aggregated to reconstruct the full set of outputs as detailed in the inference section.
For this relation, we provide 2 reasoning paths for each of 2 subject entities.
**Example**: Who won the Fields Medal? **Subgroups**: 1990 recipients and female recipients.
**Chain-of-Thought for 1990 recipients**: <think> The Fields Medal is one of the most prestigious awards in mathematics, given every four years to individuals under the age of 40 who have made outstanding contributions to the field. These are Fields Medal recipients from 1990: Vladimir Drinfeld, Vaughan Jones, Shigefumi Mori, and Edward Witten. </think> Vladimir Drinfeld, Vaughan Jones, Shigefumi Mori, Edward Witten.

For the last three relations - `hasCapacity`, `hasArea` and `companyTradesAtStockExchange` - we were unable to devise a strategy, as the triple completion task does not lend itself to reasoning-based methods. Instead, the goal is for the model to draw upon its stored knowledge about the subject entity with the hope that this will guide it towards the right answer. We show examples of CoTs used for these three relations below:

**hasCapacity.** For this relation, we provide 3 examples of human reasoning paths.
**Example**: What is the total capacity of Gaddafi Stadium?
**Chain-of-Thought** <think> Gaddafi Stadium is a major cricket stadium located in Lahore, Pakistan. Named after former Libyan leader Muammar Gaddafi, it is one of the largest cricket stadiums in Pakistan and serves as the home ground for the Pakistan national cricket team. The stadium has undergone several renovations and expansions since its construction, with its current capacity being designed to accommodate 60,000 spectators, making it one of the largest cricket venues in the world. </think> 60000

**hasArea.** For this relation, we provide 3 examples of human reasoning paths.
**Example**: What is the area of Heimaey in square kilometers?
**Chain-of-Thought** <think> Heimaey is the largest and only populated island in the Westman Islands (Vestmannaeyjar) archipelago, located off the southern coast of Iceland. Despite its small size, Heimaey is famous for being geologically active. The island has a total area of 13.4 square kilometers. </think> 13.4

**companyTradesAtStockExchange.** For this relation, we provide 3 examples of human reasoning paths.
**Example**: Where do shares of Novadios trade?
**Chain-of-Thought** <think> Novadios provides legal process outsourcing services. It is based in Los Angeles, California and operates in Argentina but it is privately held. </think> None

## 3.4. SyntheticCoT Dataset

### 3.4.1. Synthetic generation

The SyntheticCoT dataset is generated for triples from the training set using few-shot prompting with examples from HumanCoT. We use 2-shot prompting and Llama 3.3 70B Instruct as a model to distill reasoning ability from a larger model into synthetic examples. When attempting this with smaller models, we encountered frequent formatting errors, leading to an insufficiently sized SyntheticCoT dataset.
Given a triple from the training set and which does not appear in the HumanCoT dataset, this process can be repeated multiple times. Because of the stochasticity of both the generation process and the

choice of few-shot examples, the reasoning paths and answers might differ. Repeating the process more times will raise the number of correct answers and increase the diversity of reasoning paths. We repeat it 10 times in our experiments. Raising this hyperparameter will raise the size of SyntheticCoT.

**Example from the SyntheticCoT Dataset**

```
System (instructions)
Answer the question after a brief chain of reasoning. First,
write your reasoning between <think> and </think> tags. Then,
directly list the answers, separated by commas. If there’s no
answer, type "None". Be concise.
User (few-shot example from HumanCoT)
In which city did Jan Turski die?
Assistant (few-shot example from HumanCoT)
<think> Jan Turski was a Polish politician and diplomat. He was
born in 1940 and died in Warsaw in 2016. </think> Warsaw
User (few-shot example from HumanCoT)
In which city did Konrad Rufus Müller die?
Assistant (few-shot example from HumanCoT)
<think> Konrad Rufus Müller is a German photographer renowned
for his black-and-white portraits of German politicians until
Angela Merkel. I don’t have information suggesting that he has
passed away.
</think> None
User (new question)
In which city did Jeroen Brouwers die?
Assistant (generated, will be added to SyntheticCoT)
<think> Jeroen Brouwers was a Dutch journalist, writer and
critic. He passed away in 2022 in Maastricht or possibly in a
different city in the region,  but Maastricht is a known
location associated with him. </think> Maastricht
```

### 3.4.2. Filtering

**Correctness of answers.** After generating reasoning paths and answers, we discard the reasoning paths and keep only the answers. We then classify these as correct, incorrect, or incomplete. An answer is correct if it exactly matches the ground truth list. It is incomplete if it is non-empty and strictly contained within the ground truth, which is especially relevant for relations like `awardWonBy` where the true list is often long and hard to fully reproduce. Incomplete answers still carry valuable information and are retained. Numerical answers are accepted if they fall within a 5% margin of the ground truth, in line with the competition’s evaluation protocol. All other outputs are filtered out.
This filtering follows the heuristic that the reasoning is most likely correct if it leads to the correct answer.
The generation process yields 4,590 reasoning paths in total, of which 2878 are kept, 2794 are correct and 84 are incomplete. See Annex, Table 4 for per-relation details. The dataset is available in our repository[2].

[2] https://github.com/Lama-West/ReWiSe/blob/main/data/dataset2025/cot/nohelp_synthetic_cot/clean/new_synthetic_llama_correct_incomplete.csv

### 3.5. Inference on the test set

Experiments with zero-shot prompting revealed that the model often failed to follow the instructions correctly, highlighting the need for examples. In addition, when using few-shot prompting, the model should copy the reasoning strategy introduced in HumanCoT. This approach also helps establish the expected format, which is essential for processing the generated answers.

For the generation step, we use a generative large language model LLM and few-shot prompting on SyntheticCoT. $k$ is our few-shot parameter. Let's note $n_c$ the number of sampled reasoning paths. $n_c$ can be viewed as an ensembling method parameter. Raising it diminishes the risk associated to relying on a single sample. The aggregated prediction is based on a larger set of independent answers.

Given a head entity $h$ from the test set and the associated relation $r$, we do these actions $n_c$ times:

1. Select randomly $k$ elements from SyntheticCoT with relation $r$
2. Build a few-shot prompt with these elements and $(h, r)$
3. Sample from LLM to get $(CoT, a)$

This process yields $n_c$ outputs. Let's note them $(CoT_i, a_i)_{1 \leq i \leq n_c}$.

**Inference** uses $k = 5$ random few-shot examples from SyntheticCoT that share the same relation

```
System (instructions)
Answer the question after a brief chain of reasoning. First,
write your reasoning between <think> and </think> tags. Then,
directly list the answers, separated by commas. If there's no
answer, type "None". Be concise.
User (example 1, from SyntheticCoT)
In which city did Milan Lasica die?
Assistant (example 1, from SyntheticCoT)
<think> Milan Lasica was a Slovak actor, writer, and dramatist.
He died in 2021 in Bratislava. </think> Bratislava
...
User (question from test set)
In which city did Jeroen Brouwers die?
Assistant
```

### 3.6. Relation-wise self-consistency

Once the answers have been generated, we can discard the reasoning paths and move on to aggregation—this constitutes the second step of self-consistency. Standard self-consistency aggregates answers $(a_i)$ using majority voting. In our setup, however, each answer is a list of object entities $a_i = [t_{i,1}, \ldots, t_{i,j}]$, allowing us to perform aggregation at the entity level rather than at the answer level.

For one-to-many and numerical relations, we define specific aggregation methods in the next sections. For other relations expecting a single entity, such as `personHasCityOfDeath`, we simply apply majority voting over the predicted entities, including "None" as a possible outcome.

#### 3.6.1. Self-consistency for one-to-many relations

For relations that have no assumption on the length of the prediction, we define the consistency threshold $\theta \in (0, 1)$. The final prediction is the concatenation of entities which appear in at least $\theta \times n_c$ answers. Raising $\theta$ means that only tail entities consistently predicted by the model are part of the final prediction.

Instead of using an arbitrary value, we use a threshold $\theta_r$ for each relation $r$ on the validation set. This process allows for an automatic specialization of the aggregation process on relations.

For example, let's consider the triple ( USA, countryLandBordersCountry, [Canada, Mexico] ) with $n_c = 4$. This means that we use the inference model to generate four independent reasoning paths and answers. The answers are [Canada, Mexico], [Mexico], [Canada], [Canada, Panama]. Majority voting at the answer level gives a tie as no list appears twice. With a threshold $\theta = 0.5$ we keep all entities which appear in at least 2 independent answers. The aggregated answer is [Canada, Mexico].

### 3.6.2. Self-consistency for numerical relations

In the case of numerical relations, we can assume that every answer $(a_i)$ will be a numerical singleton $[t_i]$. Aggregation of all answers can be made with majority voting or with numerical aggregation methods such as using the median or the average.
The choice between these three aggregation methods is made relation-wise on the validation set.

# 4. Experiments

All experiments were conducted using Qwen3-8B. For few-shot prompting, we used 5 different examples per relation, sampled randomly from either the training set (without CoT) or from SyntheticCoT (with CoT). When Chain-of-Thought (CoT) is enabled, the model is explicitly instructed to produce reasoning enclosed between think tags before generating the final answer. We use self-consistency parameter values of 1 (no self-consistency) and 20 (with self-consistency). We first find the optimal aggregation strategy on the validation set and then apply it to the test set and report the results per relation on the test and validation sets respectively in Tables 2 and 3.

## 4.1. Relation-wise optimal aggregation strategy

This experiment is designed to find the optimal strategy for each relation. It uses the experiments with Chain-of-Thought and a number of sampled reasoning paths $n_c = 20$. Figure 1 shows the Macro-F1 score on the validation set for every strategy that is used per relation. For one-to-many relations, the strategies used correspond to the threshold $\theta$ varying between 0 and 1. For numerical strategies, we compare majority voting, and mean and median voting.

According to Figure 1, the best strategy is to use thresholds 0.05 for `awardWonBy`, 0.3 for `companyTradesAtStockExchange` and 0.5 for `countryLandBordersCountry`. For `personHasCityOfDeath`, there cannot be multiple answers, so answers are aggregated using majority voting and considering None as a valid proposition. According to Figure 1, the best strategy is to use the median for `hasArea`, and majority voting for `hasCapacity`. In our submission however, we fixed `hasCapacity` to median based on earlier experiments. Median also appears stronger when looking across consistency levels, not only at level 20. Since both strategies are close in performance, this difference has little impact on the overall results.
The strategy analysis without CoT can be found in the Annex (Figure 2).

## 4.2. Results split per relation

Table 2 compares our best method with those of the official baseline on the test set. Macro scores are reported for each relation. The baseline uses a single direct generation with no Chain-of-Thought. Our method uses Chain-of-Thought prompting and self-consistency ($n_c = 20$). When self-consistency is used, the optimal strategy as detailed right above is used. Our approach consistently improves Macro-F1 across all relations except `awardWonBy`.

Table 3 breaks down the validation set results by relation type, comparing inference with and without Chain-of-Thought (CoT) reasoning, as well as with self-consistency sampling ($n_c = 20$) and without it

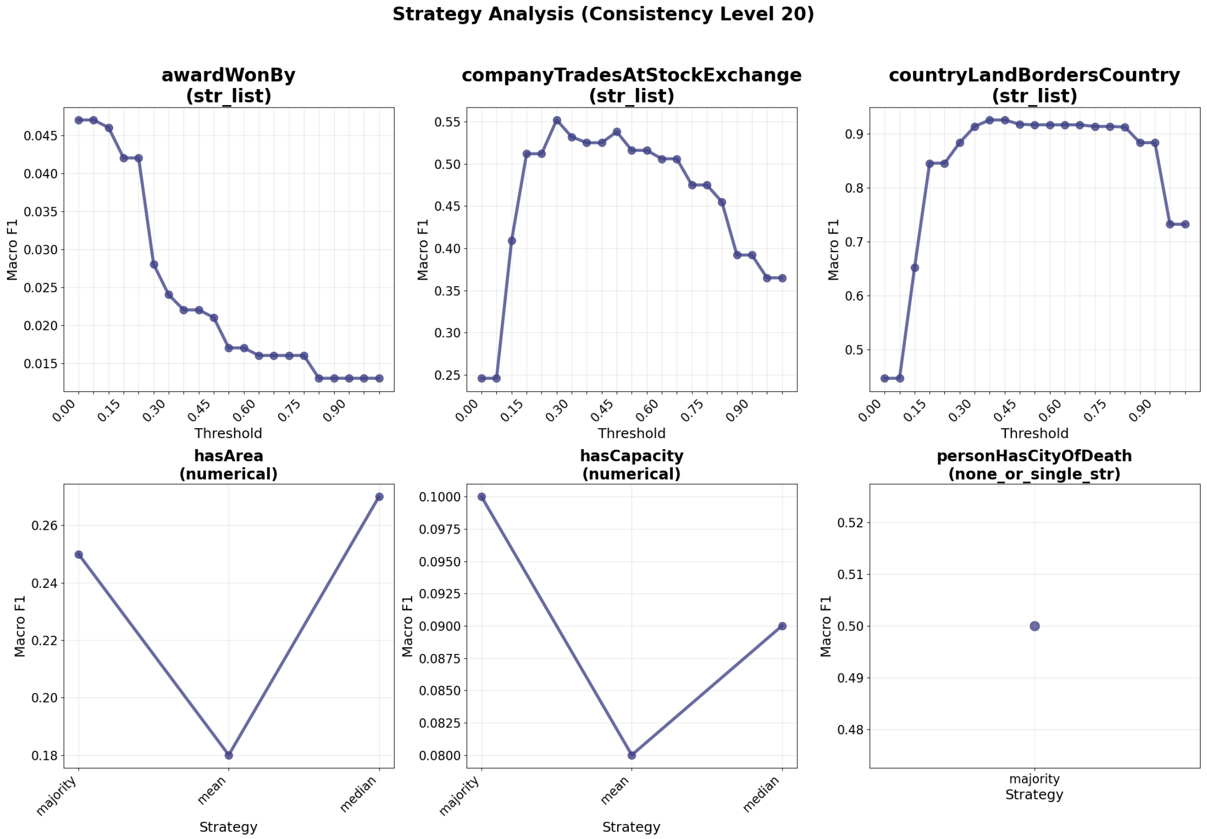

**Figure 1:** Relation-wise comparison of aggregation strategies using 20 independent predictions on the validation set **with CoT**

**Table 2**
Comparison of macro precision, recall, and F1 score on the test set. Our method combines CoT prompting with self-consistency ($n_c = 20$). The baseline uses direct generation ($n_c = 1$), without CoT.

| Relation | Baseline (No CoT, $n_c = 1$) | | | Ours (CoT, $n_c = 20$) | | |
|---|---|---|---|---|---|---|
| | Macro-p | Macro-r | Macro-F1 | Macro-p | Macro-r | Macro-F1 |
| awardWonBy | 0.240 | 0.090 | **0.117** | 0.054 | 0.193 | 0.083 |
| companyTradesAtStockExchange | 0.185 | 0.591 | 0.167 | 0.670 | 0.629 | **0.543** |
| countryLandBordersCountry | 0.768 | 0.813 | 0.703 | 0.924 | 0.894 | **0.890** |
| hasArea | 0.240 | 0.240 | 0.240 | 0.320 | 0.320 | **0.320** |
| hasCapacity | 0.040 | 0.040 | 0.040 | 0.140 | 0.090 | **0.090** |
| personHasCityOfDeath | 0.080 | 0.650 | 0.080 | 0.850 | 0.640 | **0.560** |
| **All Relations** | 0.227 | 0.435 | 0.212 | 0.546 | 0.482 | **0.444** |

($n_c = 1$). The experiments are compared with macro precision, recall and F1 score. The highest score per relation is in **bold**. When self-consistency is used, the optimal strategy is used.

An analysis of Tables 2 and 3 highlights several notable points:

- Self-consistency ($n_c = 20$) convincingly improves performance over predictions with a single generation ($n_c = 1$). This is consistently the case when using Chain-of-Thought reasoning but also when not using it.
- Chain-of-Thought reasoning yields mixed effects. For some relations like `countryLandBordersCountry` and `companyTradesAtStockExchange`, CoT combined with self-consistency provides the highest Macro-F1. However, for relations like `awardWonBy` and `personHasCityOfDeath`, CoT does not always improve—and sometimes slightly

**Table 3**
Macro results on the validation set detailed by relation, CoT usage, and consistency noted as $n_c$. All experiments use Qwen3-8b. When not using CoT, the few-shot examples are taken at random from the training set. When using CoT, the model is prompted to use Chain-of-Thought reasoning and the few-shot examples are taken at random from SyntheticCoT. For all experiments, few-shot prompting is done with 5 examples that share the same relation.
$n_c$ is the self-consistency parameter which is the number of independent answers generated for each element of the inference dataset, here the validation set. When $n_c = 1$, one answer is generated and used as a prediction. When $n_c > 1$, aggregation is done with the best method for the relation.

| Relation | CoT | $n_c$ | Macro-p | Macro-r | Macro-F1 |
|---|---|---|---|---|---|
| awardWonBy | No | 1 | 0.155 | 0.039 | 0.055 |
| awardWonBy | No | 20 | 0.615 | 0.064 | **0.068** |
| awardWonBy | Yes | 1 | 0.166 | 0.006 | 0.011 |
| awardWonBy | Yes | 20 | 0.156 | 0.038 | 0.046 |
| companyTradesAtStockExchange | No | 1 | 0.241 | 0.625 | 0.246 |
| companyTradesAtStockExchange | No | 20 | 0.656 | 0.643 | 0.533 |
| companyTradesAtStockExchange | Yes | 1 | 0.200 | 0.541 | 0.189 |
| companyTradesAtStockExchange | Yes | 20 | 0.757 | 0.638 | **0.564** |
| countryLandBordersCountry | No | 1 | 0.550 | 0.877 | 0.574 |
| countryLandBordersCountry | No | 20 | 0.807 | 0.922 | 0.829 |
| countryLandBordersCountry | Yes | 1 | 0.639 | 0.886 | 0.628 |
| countryLandBordersCountry | Yes | 20 | 0.897 | 0.950 | **0.905** |
| hasArea | No | 1 | 0.120 | 0.120 | 0.120 |
| hasArea | No | 20 | 0.190 | 0.180 | 0.180 |
| hasArea | Yes | 1 | 0.230 | 0.230 | 0.230 |
| hasArea | Yes | 20 | 0.260 | 0.260 | **0.260** |
| hasCapacity | No | 1 | 0.030 | 0.030 | 0.030 |
| hasCapacity | No | 20 | 0.170 | 0.080 | 0.080 |
| hasCapacity | Yes | 1 | 0.020 | 0.020 | 0.020 |
| hasCapacity | Yes | 20 | 0.110 | 0.090 | **0.090** |
| personHasCityOfDeath | No | 1 | 0.070 | 0.520 | 0.070 |
| personHasCityOfDeath | No | 20 | 0.880 | 0.530 | **0.500** |
| personHasCityOfDeath | Yes | 1 | 0.075 | 0.530 | 0.077 |
| personHasCityOfDeath | Yes | 20 | 0.790 | 0.510 | 0.450 |
| **All Relations** | No | 1 | 0.178 | 0.396 | 0.180 |
| **All Relations** | No | 20 | 0.524 | 0.432 | 0.390 |
| **All Relations** | Yes | 1 | 0.204 | 0.402 | 0.197 |
| **All Relations** | Yes | 20 | 0.532 | 0.449 | **0.415** |

reduces—performance compared to direct generation.

- Numerical relations (`hasArea`, `hasCapacity`) achieve modest gains with CoT and self-consistency. Their absolute Macro-F1 remains low, highlighting the difficulty of these relations.
- Overall performance is highest when combining CoT with self-consistency ($n_c = 20$), reaching a Macro-F1 score of 44.4%, beating the baseline (21.2%) by 23.2 points on the test set.

# 5. Discussion and Limitations

## 5.1. What if the model doesn't know the answer?

For some training triples, the model consistently fails to produce a correct answer during the construction of SyntheticCoT. This raises the issue of what the model actually knows, and whether attempting to retrieve certain knowledge from its parametric memory is meaningful. Since research on interpreting and extracting knowledge from LLM weights is ongoing, filtering out information the model clearly doesn't know could help reduce reliance on guesses. In our approach, we address this by removing incorrect answers from SyntheticCoT: any triple the model never answers correctly during SyntheticCoT generation is excluded from the prompts used at inference time. As SyntheticCoT is generated with a larger model, this strategy assumes that the knowledge of the smaller Qwen3-8B model is contained within that of the larger Llama3.3 70B model.

### 5.2. $\theta$ as a way to control the precision / recall tradeoff

The consistency threshold $\theta$ is a hyperparameter controlling the precision / recall tradeoff. As an example, triples involving the relation `awardWonBy` expect many object entities. The challenge is thus to make many propositions, keeping entities that are rarely predicted, which corresponds to setting a low threshold. The recall is indeed lower than the precision in our experiments on the validation set (see table 3). In this case, the optimal threshold was 0.05 (see Figure 1) which means that all entities that appear in at least $5\%$ of independent answers are kept in the prediction.

The strategy in HumanCoT for this relation was to predict only a subgroup of tail entities and rely on self-consistency to aggregate different subgroups. This is consistent with fixing a low threshold.

We further analyzed the effect of increasing the number of self-consistency samples (see Annex, Fig. 3) and observed that gains saturate quickly, typically before $n_c = 20$, with the ceiling varying by relation.

### 5.3. Sampling few-shot examples from SyntheticCoT

We also tested whether self-consistency remains effective when the few-shot examples are fixed across generations. In this set-up, as we got rid of the stochasticity in the choice of few-shot examples, the stochasticity comes exclusively from the generation process. Using a reduced SyntheticCoT of five random examples per relation, performance averaged 0.398 Macro-F1 on the validation set (across seven runs), compared to 0.415 when sampling from the full SyntheticCoT dataset. This shows that self-consistency benefits from diversity in few-shot examples. The generation process still provides enough stochasticity to induce a large improvement over single-sample inference which gets 19.7.

### 5.4. Limitations

There are a few limitations to our work.

- First, the method still depends on a human annotator to write HumanCoT examples, and the quality of this dataset is crucial to the overall performance. In our case, it was constructed based on heuristics. Exploring ways to optimize the construction of HumanCoT and assessing the scalability of the method—particularly whether HumanCoT can be effectively applied to unseen relations—would be valuable directions for future work.
- Second, the way we build SyntheticCoT assumes that a CoT is correct if the answer is correct. This is not always the case, especially when there is no correct answer. A better filtering would be beneficial to have a cleaner SyntheticCoT.
- Finally, we still have not found the ideal solution for relations such as `awardWonBy` and `hasCapacity`. Whether the model weights actually contain the required answers is an open question.

## 6. Conclusion

In this paper, we propose a Relation-Wise Self-consistency, a method that uses a very limited amount of human-written Chains-of-Thought to build SyntheticCoT, a larger and synthetic set of reasoning paths that lead to a correct answer. This dataset is used in a few-shot prompting fashion to give a systematic way to reason about given relations in order to complete triples. We then adapt self-consistency to a general setting without making assumptions about the cardinality of the relations, instead tuning the prediction aggregation process on a per-relation basis. Our results show that our method ReWiSe improves performance on the LM-KBC 2025 challenge, achieving a Macro-F1 score of 44.4% on the test set, a gain of 23.2 points over the baseline.

Overall, this year's edition of the LM-KBC challenge did not allow fine-tuning. However, training the LLM to learn relations instead of few-shot prompting could be an interesting future work. It could take advantage of our SyntheticCoT dataset. The intuition for self-consistency is that multiple reasoning paths can lead to the correct answer. Training on SyntheticCoT would take advantage of the multiplicity of reasoning paths that lead to the correct answer for a given relation.

## Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

## Acknowledgments

We thank NSERC for supporting this work through a Discovery Grant and Compute Canada and Calcul Québec for the computational resources provided.

## References

[1] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, J. Tang, GPT Understands, Too, 2023. URL: http://arxiv.org/abs/2103.10385. doi:10.48550/arXiv.2103.10385, arXiv:2103.10385 [cs].

[2] M. Zhang, O. Press, W. Merrill, A. Liu, N. A. Smith, How Language Model Hallucinations Can Snowball, 2023. URL: http://arxiv.org/abs/2305.13534. doi:10.48550/arXiv.2305.13534, arXiv:2305.13534 [cs].

[3] R. Zhao, A. Köksal, A. Modarressi, M. A. Hedderich, H. Schütze, Do We Know What LLMs Don't Know? A Study of Consistency in Knowledge Probing, 2025. URL: http://arxiv.org/abs/2505.21701. doi:10.48550/arXiv.2505.21701, arXiv:2505.21701 [cs].

[4] C. Jiang, B. Qi, X. Hong, D. Fu, Y. Cheng, F. Meng, M. Yu, B. Zhou, J. Zhou, On Large Language Models' Hallucination with Regard to Known Facts, 2024. URL: http://arxiv.org/abs/2403.20009. doi:10.48550/arXiv.2403.20009, arXiv:2403.20009 [cs].

[5] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, H. Hajishirzi, When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories, 2023. URL: http://arxiv.org/abs/2212.10511. doi:10.48550/arXiv.2212.10511, arXiv:2212.10511 [cs].

[6] J.-C. Kalo, T.-P. Nguyen, S. Razniewski, B. Zhang, Lm-kbc challenge @ iswc 2025, in: 4th Semantic Web Challenge on Language Models for Knowledge Base Construction Challenge, 2025. URL: https://lm-kbc.github.io/challenge2025/.

[7] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85. URL: https://dl.acm.org/doi/10.1145/2629489. doi:10.1145/2629489.

[8] B. Zhang, I. Reklos, N. Jain, A. M. Peñuela, E. Simperl, Using Large Language Models for Knowledge Engineering (LLMKE): A Case Study on Wikidata, 2023. URL: https://arxiv.org/abs/2309.08491. doi:10.48550/ARXIV.2309.08491, version Number: 1.

[9] L. Wang, W. Zhao, Z. Wei, J. Liu, SimKGC: Simple Contrastive Knowledge Graph Completion with Pre-trained Language Models, 2022. URL: http://arxiv.org/abs/2203.02167, arXiv:2203.02167 [cs].

[10] 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) 1–20. URL: https://ieeexplore.ieee.org/document/10387715/. doi:10.1109/TKDE.2024.3352100.

[11] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, S. Riedel, Language Models as Knowledge Bases? (2019). URL: https://arxiv.org/abs/1909.01066. doi:10.48550/ARXIV.1909.01066, publisher: [object Object] Version Number: 2.

[12] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, S. Singh, AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts, 2020. URL: http://arxiv.org/abs/2010.15980. doi:10.48550/arXiv.2010.15980, arXiv:2010.15980 [cs].

[13] X. L. Li, P. Liang, Prefix-Tuning: Optimizing Continuous Prompts for Generation, 2021. URL: http://arxiv.org/abs/2101.00190. doi:10.48550/arXiv.2101.00190, arXiv:2101.00190 [cs].

[14] S. Pitis, M. R. Zhang, A. Wang, J. Ba, Boosted Prompt Ensembles for Large Language Models, 2023. URL: http://arxiv.org/abs/2304.05970. doi:10.48550/arXiv.2304.05970, arXiv:2304.05970 [cs].

[15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2023. URL: http://arxiv.org/abs/2201.11903, arXiv:2201.11903 [cs].

[16] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, Z. Qiu, Qwen3 Technical Report, 2025. URL: http://arxiv.org/abs/2505.09388. doi:10.48550/arXiv.2505.09388, arXiv:2505.09388 [cs].

[17] D. Su, M. Patwary, S. Prabhumoye, P. Xu, R. Prenger, M. Shoeybi, P. Fung, A. Anandkumar, B. Catanzaro, Context Generation Improves Open Domain Question Answering, 2023. URL: http://arxiv.org/abs/2210.06349. doi:10.48550/arXiv.2210.06349, arXiv:2210.06349 [cs].

[18] Z. Jiang, F. F. Xu, J. Araki, G. Neubig, How Can We Know What Language Models Know?, 2020. URL: http://arxiv.org/abs/1911.12543. doi:10.48550/arXiv.1911.12543, arXiv:1911.12543 [cs].

[19] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. El-Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, J. Kaplan, Language Models (Mostly) Know What They Know, 2022. URL: http://arxiv.org/abs/2207.05221. doi:10.48550/arXiv.2207.05221, arXiv:2207.05221 [cs].

[20] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, D. Zhou, Self-Consistency Improves Chain of Thought Reasoning in Language Models, 2023. URL: http://arxiv.org/abs/2203.11171. doi:10.48550/arXiv.2203.11171, arXiv:2203.11171 [cs].

[21] X. Chen, R. Aksitov, U. Alon, J. Ren, K. Xiao, P. Yin, S. Prakash, C. Sutton, X. Wang, D. Zhou, Universal Self-Consistency for Large Language Model Generation, 2023. URL: http://arxiv.org/abs/2311.17311. doi:10.48550/arXiv.2311.17311, arXiv:2311.17311 [cs].

[22] Y. Wang, S. Zhao, Z. Wang, H. Huang, M. Fan, Y. Zhang, Z. Wang, H. Wang, T. Liu, Strategic Chain-of-Thought: Guiding Accurate Reasoning in LLMs through Strategy Elicitation, 2024. URL: http://arxiv.org/abs/2409.03271. doi:10.48550/arXiv.2409.03271, arXiv:2409.03271 [cs].

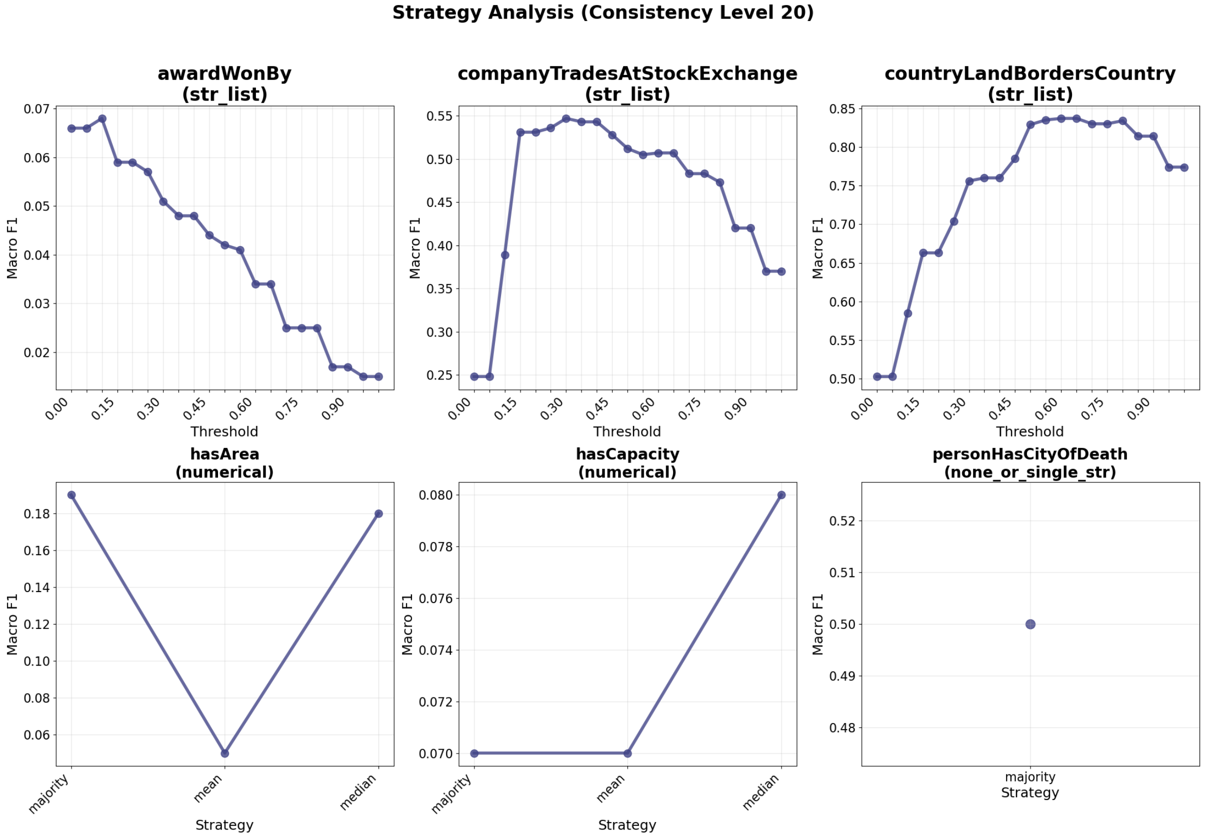

**Figure 2:** Relation-wise comparison of aggregation strategies using 20 independent predictions **without CoT**

# 7. Annex

## 7.1. Relation-wise optimal aggregation strategy

Raising $n_c$ is an ensembling method and it diminishes the risk as the prediction uses a larger set of independent answers. The prediction is relying less on a single answer as $n_c$ grows. For both experiments with and without CoT, and using $n_c = 20$, the scores of the different aggregation strategies for each relation on the validation set are shown in Figures 1 and 2, respectively.

Figure 3 illustrates the impact of raising the number of self-consistency samples $n_c$ from 1 up to 100 for each relation on the validation set Macro-F1 score. The curves show that performance generally increases quickly at small $n_c$ (up to around 20), after which it plateaus. Most of the benefit of relation-wise self-consistency can thus be achieved with relatively few samples, although the exact point of saturation depends on the relation. For example, `personHasCityOfDeath` continues to improve until around $n_c = 20$, while `companyTradesAtStockExchange` reaches its plateau earlier (around $n_c = 10$). In contrast, `hasArea` remains low overall, indicating limited benefit from self-consistency for this relation.

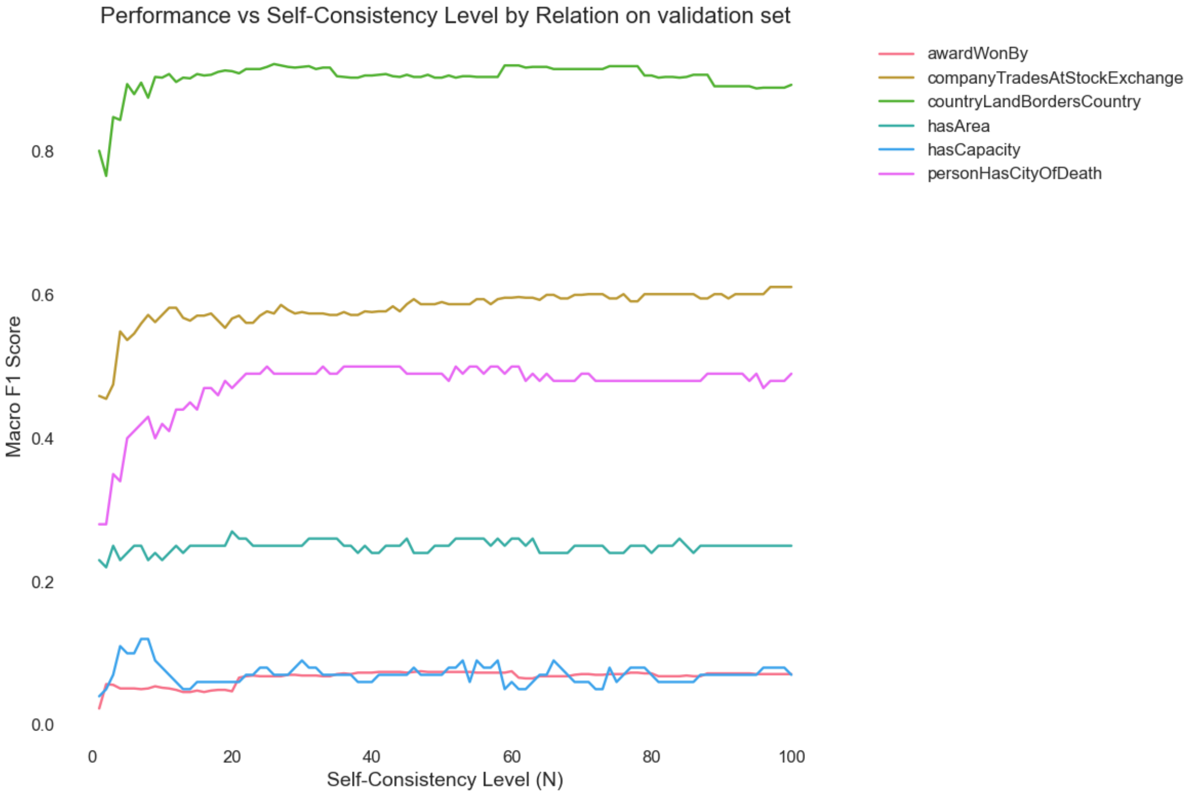

**Figure 3:** Macro-F1 score as a function of self-consistency level $n_c$, on the validation set, split by relation. To compute the prediction at self-consistency level $n$, we keep the first $n$ predictions over $100$.

## 7.2. SyntheticCoT post-processing

Table 4 shows the statistics of the filtering process for Llama 70B. Incomplete outputs are particularly important to keep for relation `awardWonBy` as there is no correct generated output. This is due to the difficulty of generating the full exact list for relations with high cardinality.

**Table 4**
Relation Statistics for Llama 70B Model, dataset 2025. SyntheticCoT keeps the correct and incomplete outputs. Its size is 2878.

| Relation | Total | Correct | % Correct | Incomplete |
|---|---:|---:|---:|---:|
| hasArea | 980 | 664 | 67.8% | 0 |
| hasCapacity | 970 | 440 | 45.4% | 0 |
| countryLandBordersCountry | 630 | 578 | 91.7% | 19 |
| personHasCityOfDeath | 960 | 571 | 59.5% | 0 |
| companyTradesAtStockExchange | 970 | 541 | 55.8% | 39 |
| awardWonBy | 80 | 0 | 0.0% | 26 |
| Overall | 4590 | 2794 | 60.9% | 84 |