

Titre: Knowledge Graph Construction from Pre-Trained Language Models
Title:

Auteur: Edouard Albert-Roulhac
Author:

Date: 2026

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Albert-Roulhac, E. (2026). Knowledge Graph Construction from Pre-Trained Language Models [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie.
Citation: <https://publications.polymtl.ca/73223/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/73223/>
PolyPublie URL:

**Directeurs de
recherche:** Amal Zouaq
Advisors:

Programme: Génie Informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Knowledge Graph Construction from Pre-Trained Language Models

EDOUARD ALBERT-ROULHAC

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie informatique

Février 2026

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

Knowledge Graph Construction from Pre-Trained Language Models

présenté par **Edouard ALBERT-ROULHAC**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Michel DESMARAIS, président

Amal ZOUAQ, membre et directrice de recherche

Guillaume-Alexandre BILODEAU, membre

DEDICATION

*To my family, my friends, and all those I met on the path,
may our adventure continue...*

ACKNOWLEDGEMENTS

I would first like to express my sincere gratitude to my supervisor, Amal Zouaq, for her guidance, trust, and support throughout this research.

I am deeply thankful to all the members of the LAMA–WeST laboratory for their collaboration, feedback, and the friendly atmosphere that made this work so rewarding, and I wish them all the very best, in particular Djalil for continuing this line of research.

I would also like to acknowledge the Mila community, Polytechnique Montréal, and Télécom Paris for providing such stimulating research environments and opportunities, as well as the LM-KBC challenge organizers for creating an open and constructive benchmark that motivated this work.

This research was supported by NSERC through a Discovery Grant, and benefited from the computational resources provided by Compute Canada and Calcul Québec.

Finally, I would like to thank my family and friends for their patience, encouragement, and constant support.

RÉSUMÉ

Les graphes de connaissances offrent une représentation utile de la connaissance générale, mais ils sont incomplets. Les grands modèles de langue ont des connaissances générales qu'ils ont apprises sur une grande quantité de texte non structuré, mais ils les stockent sous forme paramétrique, ce qui la rend difficile à exploiter. Ce mémoire étudie l'extraction de cette connaissance paramétrique, sans ressource extérieure, afin de construire ou compléter un graphe de connaissances. Le problème est formulé dans le cadre de la compétition LM-KBC qui défie les participants d'extraire le mieux possible certaines connaissances d'un modèle de langue. Notre solution ReWiSe produit plusieurs raisonnements et réponses puis les agrège en tenant compte de connaissances spécifiques à cette relation ainsi que des contraintes du graphe.

Nous introduisons en premier lieu HumanCoT, un petit ensemble de données avec des raisonnements rédigés à la main et spécifiques aux relations. Utilisés en prompt pour guider le modèle, ces exemples permettent de générer SyntheticCoT, un ensemble plus large de raisonnements systématiques conduisant aux bonnes réponses, qui sert d'exemples pour l'inférence sur le jeu de test. Par la suite, nous adaptons l'auto-cohérence aux schémas des graphes de connaissances. Pour effectuer une requête, nous produisons plusieurs générations et agrégeons les sorties au niveau des entités avec des stratégies par relation pour respecter leur schéma et cardinalité.

Évalué sur LM-KBC 2025 avec Qwen3-8B, sans accès à des données externes ni affinage du modèle, ReWiSe remporte la compétition. Par la suite, au-delà du prompting, nous explorons les méthodes d'entraînement partiel (PEFT) sur SyntheticCoT afin de tester si, malgré le peu de données d'entraînement, les méthodes usuelles d'entraînement améliorent la capacité à récupérer des connaissances dans les paramètres des modèles ; les améliorations sont inégales selon les relations.

Nous publions le code complet de ReWiSe ainsi que les jeux de données HumanCoT et SyntheticCoT afin de favoriser la reproductibilité et de soutenir des travaux ultérieurs. Les résultats montrent que de petits patrons de raisonnement adaptés au schéma, combinés à une agrégation relationnelle au moment de l'inférence, suffisent à extraire de façon fiable des faits structurés de la mémoire paramétrique d'un modèle, constituant ainsi une référence qui n'utilise pas de données externes pour la construction de graphes de connaissances.

ABSTRACT

Knowledge graphs are a precise, general but incomplete way to structure knowledge; large language models have learnt broad world knowledge on unstructured data but this knowledge is parametric thus hard to extract. This thesis studies how to read out that parametric knowledge to construct or complete a knowledge graph. We frame the problem under the LM-KBC rules and propose ReWiSe, a relation-wise self-consistency method that combines relation-specific probing with schema-aware test-time scaling to better exploit the stochasticity of LLM text generation.

First, we introduce HumanCoT, a tiny set of hand-written, relation-specific reasoning patterns. It is used in few-shot fashion to generate SyntheticCoT, a larger set of systematic reasoning paths leading to correct answers. This dataset serves as prompting examples for test-time generation. Then, we adapt self-consistency to knowledge-graph schemas: we produce multiple generations per head-relation query and aggregate answers at the entity-level relation-wise strategies.

Evaluated on LM-KBC 2025 with the imposed Qwen3-8B model and no external retrieval or full-model fine-tuning, ReWiSe wins the challenge. Beyond prompting, we explore parameter-efficient fine-tuning on SyntheticCoT to test whether light training on scarce data yields gains on LLM knowledge probing; gains are uneven across relations.

We release the complete ReWiSe codebase together with HumanCoT and SyntheticCoT to support reproducibility and further research. The results show that our entity-level and relation-wise adaptation of self-consistency aggregation is beneficial to structured knowledge extraction from an LLM’s parametric memory, offering a strong closed-book baseline for knowledge base construction.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vi
LIST OF TABLES	x
LIST OF FIGURES	xii
LIST OF SYMBOLS AND ACRONYMS	xiii
LIST OF APPENDICES	xiv
CHAPTER 1 INTRODUCTION	1
1.1 General Context	1
1.2 LM-KBC	3
1.3 Research Objectives	4
1.4 Thesis Outline	5
CHAPTER 2 BACKGROUND AND LITERATURE REVIEW	7
2.1 Large Language Models	7
2.1.1 Pre-training and Parametric Memory	7
2.1.2 In-context learning	8
2.1.3 Test-time scaling	9
2.1.4 Fine-tuning	9
2.2 Knowledge Graphs	11
2.2.1 Formalization	11
2.2.2 Automatic construction of Knowledge Graphs	12
2.2.3 Knowledge Graph Completion	13
2.2.4 Evaluation and Pitfalls	14
2.2.5 Knowledge Graphs and Language Models	15
2.3 LLM Knowledge Probing	16
2.3.1 Prompt-based probing	16
2.3.2 Test-time scaling for knowledge probing	16

2.4	Summary and open challenges	18
CHAPTER 3 LANGUAGE MODEL PROBING METHODS FOR KNOWLEDGE BASE CONSTRUCTION		20
3.1	The LM-KBC competition	20
3.1.1	Datasets	20
3.1.2	Competition rules	21
3.1.3	Evaluation and metrics	23
3.1.4	Baseline and limitations	24
3.2	Relation-Wise Self-consistency	25
3.2.1	HumanCoT dataset	26
3.2.2	SyntheticCoT Dataset	29
3.2.3	Inference on the test set	32
3.2.4	Relation-wise self-consistency	33
3.3	Sub-queries for high-cardinality relations	37
3.3.1	Motivation and formalization	37
3.3.2	Implementation	37
3.3.3	Example	40
3.4	Relation Tuning	42
3.4.1	Motivation	42
3.4.2	Training	42
3.4.3	Inference	44
3.5	Overview of the presented methods	44
CHAPTER 4 EXPERIMENTS AND DISCUSSION		45
4.1	Experimental setup	45
4.1.1	Datasets	45
4.1.2	Models and hyperparameters	45
4.1.3	System prompts	46
4.1.4	SyntheticCoT construction, experimental details	46
4.1.5	Relation Tuning experimental setup	48
4.2	Experiments	49
4.2.1	Optimal aggregation strategy per relation	49
4.2.2	ReWiSe ablations: Chain-of-Thought and self-consistency	51
4.2.3	Sub-queries for <code>awardWonBy</code>	56
4.2.4	Relation Tuning	60
4.2.5	Experiments on the unified dataset	63

4.3	Discussion	67
4.3.1	The role of chain-of-thought reasoning	67
4.3.2	Few-shot prompts and soft prompts	69
4.3.3	Pros and cons of training	69
4.3.4	Compute cost	70
4.3.5	High-cardinality relations and query partitioning	70
4.3.6	Cross-dataset generalization	71
4.3.7	Knowledge limitations	71
4.3.8	SyntheticCoT quality concerns	72
CHAPTER 5 CONCLUSION		73
5.1	Summary of Contributions	73
5.2	Limitations	73
5.3	Future Research	74
REFERENCES		75
APPENDICES		88

LIST OF TABLES

Table 3.1	Unified dataset: per-relation subject counts across splits	22
Table 3.2	Dataset 2025 baseline: relation-to-question templates	25
Table 3.3	Relation Statistics for Llama 70B Model. The SyntheticCoT dataset consists of both correct and incomplete outputs, totaling 2878 examples.	30
Table 3.4	Comparison between Relation Tuning training data sources	44
Table 4.1	Per-relation aggregation strategy on Dataset2025 selected on the validation set for the Chain-of-Thought (CoT) and no-CoT regimes with $n_c = 20$	51
Table 4.2	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 CoT 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. 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.	54
Table 4.3	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. In this case, the baseline is the official run from the competition organizers.	55
Table 4.4	LM-KBC 2025 final leaderboard (test set Macro-F1).	57
Table 4.5	Performance on awardWonBy relation (validation set) comparing baseline, ReWiSe, and Sub-queries approaches. All experiments use Qwen3-8B with 5-shot prompting. Baseline uses $n_c = 1$ (single generation). ReWiSe uses $n_c = 20$ with self-consistency on identical questions. Sub-queries uses $n_c = 20$ with diverse question formulations generated by Qwen3-8B, aggregating predictions across different phrasings of the same query.	59

Table 4.6	Per-relation Macro-F1 comparison on dataset2025 validation set between ReWiSe and Relation Tuning with different prompting strategies. All experiments use Qwen3-8B with $n_c = 20$ (self-consistency). ReWiSe uses the base model with 5-shot prompting. RT 0-shot uses the trained model (10 virtual tokens, 10 epochs) with zero-shot inference. RT 5-shot combines the trained model with 5-shot prompting at inference time. Best result for each relation is in bold and second best is in <i>italic</i>	62
Table 4.7	Relation Statistics for the unified SyntheticCoT dataset. It consists CoTs generated by Llama3.3 70B on train examples of the unified dataset. It uses 2-shot prompting over HumanCoT examples that don't share the same relation. CoTs leading to correct or incomplete outputs are kept, totaling 12,420 examples across 27 relations.	65
Table 4.8	Macro-F1 performance on <code>dataset_all</code> validation set comparing baseline, ReWiSe, and Relation Tuning approaches with and without CoT. All experiments use Qwen3-8B. Prompting experiments (ReWiSe, Baseline, one-prediction CoT) use 5-shot prompting while Relation Tuning uses zero-shot prompting. Baseline uses $n_c = 1$ (single generation). ReWiSe uses $n_c = 20$ with self-consistency. Relation Tuning uses prompt tuning with 10 virtual tokens trained for 10 epochs, then applies $n_c = 20$ self-consistency at inference. Results shown are from the tenth epoch.	66

LIST OF FIGURES

Figure 1.1	Example fragment of a knowledge graph. Each directed edge represents a relation r from subject h to object t in a triple $\langle h, r, t \rangle$	2
Figure 4.1	Relation-wise comparison of aggregation strategies using 20 independent predictions on the validation set with CoT . In red, the best strategy on the validation set. This strategy is chosen for the prediction.	52
Figure 4.2	Relation-wise comparison of aggregation strategies using 20 independent predictions on the validation set without CoT . In red, the best strategy on the validation set. This strategy is chosen for the prediction.	53
Figure 4.3	Performance difference (CoT - No CoT) in Macro-F1 points difference for each relation using ReWiSe on the unified dataset validation set. Relations are ordered by descending difference.	67
Figure 4.4	Performance difference (CoT - No CoT) in Macro-F1 points difference for each relation using Relation Tuning on the unified dataset validation set. Relations are ordered by descending difference.	68
Figure A.1	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.	88

LIST OF SYMBOLS AND ACRONYMS

long-plural-form=Chains-of-Thoughtshort-plural-form=CoTs

KG	Knowledge Graph
LLM	Large Language Model
PLM	Pre-trained Language Model
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
CoT	Chain-of-Thought
RAG	Retrieval-Augmented Generation
LM-KBC	Language Model for Knowledge Base Construction

LIST OF APPENDICES

Appendix A	Influence of the self-consistency level	88
------------	---	----

CHAPTER 1 INTRODUCTION

1.1 General Context

Knowledge Graphs (KGs) are a useful way to structure and represent knowledge. They are flexible enough to encode general world knowledge and are widely used for tasks such as question answering, search, recommendation, and analytics [1]. A KG is a set of triples $\langle h, r, t \rangle$ where h is the head entity, r the relation, and t the tail entity. Figure 1.1 illustrates a small subset of a knowledge graph. Building large, multi-domain KGs is a long and difficult process. Resources like Wikidata combine community curation, which is difficult to scale, with automated extraction from text or tables, which depends on high-quality upstream sources. Despite their scale, world KGs remain incomplete, since enumerating all true triples is not feasible. This motivates research on automatic KG construction and completion.

Exploiting knowledge from Pre-trained Language Models (PLMs) is a promising path. During pre-training on internet-scale corpora with a self-supervised objective (next-token or masked-token prediction), PLMs acquire broad factual and commonsense knowledge that is stored in their parameters [2]. They do not require curated, structured databases at training time and can therefore be seen as automatically built, general-purpose repositories of world knowledge and therefore as potential sources of knowledge. It is natural to attempt to extract and structure that knowledge into a classical knowledge base, and in particular into a KG. Once organized, knowledge can be efficiently retrieved and integrated into downstream applications. Extracting structured knowledge from PLMs also opens the study of the knowledge these models learn.

However, the knowledge of Large Language Models (LLMs) is distributed in the model’s parameters and is not directly human-interpretable. Probing can elicit a significant fraction of relational facts, but generation is stochastic and sensitive to phrasing, so a model may produce inconsistent answers to semantically equivalent inputs. In addition, LLMs are prone to hallucinations, and pre-training corpora are often undisclosed, which complicates provenance. Determining what an LLM knows, and with what reliability, remains an active research area [3–5]. Altogether, using LLMs as knowledge sources requires methods that improve robustness to prompt variance and stochastic decoding, reduce hallucinations, and respect KG schema constraints as well as relation cardinalities.

One attractive option is to treat the pre-trained LLM as a fixed knowledge store: leaving its parameters untouched avoids directly editing, and potentially harming, its internal memory.

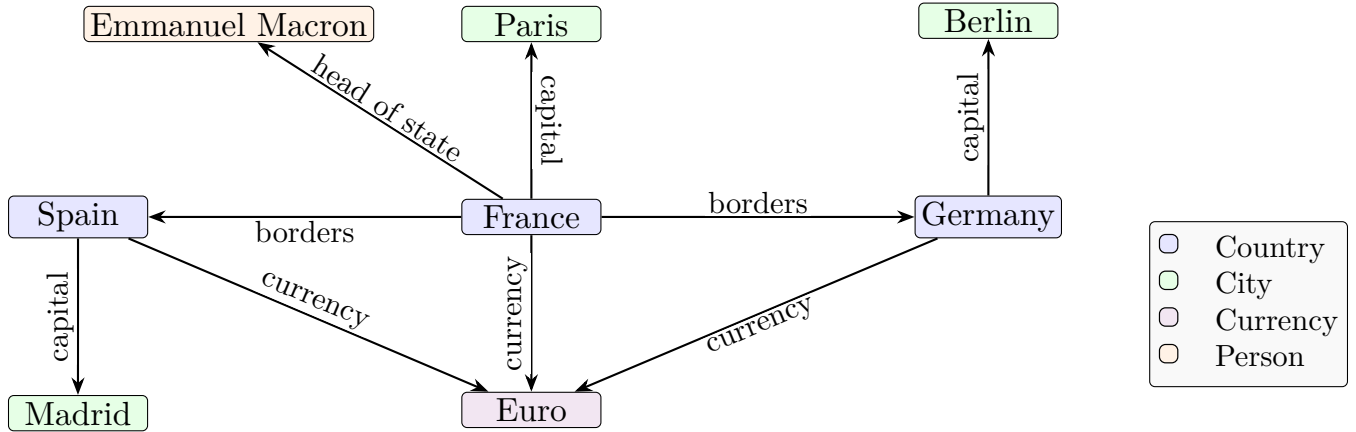


Figure 1.1 Example fragment of a knowledge graph. Each directed edge represents a relation r from subject h to object t in a triple $\langle h, r, t \rangle$.

In that regime, we aim to increase performance by spending more computation at inference time in order to make better use of the same parametric knowledge without modifying it. This can be done by scaling compute in parallel using self-consistency: sampling multiple candidate answers and aggregating them into a final prediction. This aggregation reduces the risk associated with relying on a single prediction that is the result of a stochastic process. Querying the model multiple times and comparing the answers is a way to have more information on the model’s knowledge. A complementary strategy increases computation sequentially, by letting the model produce intermediate reasoning steps before giving an answer. If the reasoning process helps the model retrieve its parametric knowledge, then the performance can increase on knowledge-heavy tasks. Such reasoning paths can help the model reach better final predictions, but this type of guidance requires high-quality, relation-specific supervision describing how to reason (rather than only what the answer is). This kind of data is not always available, which motivates the use of synthetic reasoning traces tailored to the KG.

A complementary way to proceed modifies the model itself. Fine-tuning on task-specific data is the standard way to improve performance and guide behavior on most downstream tasks, but it typically assumes abundant, carefully curated supervision, with high-quality data that is also not necessarily available. Using training methods can also harm the parametric memory of the model directly so a training method to probe LLMs should be careful about which parameters it updates.

1.2 LM-KBC

We study closed-book structured knowledge extraction from a given PLM, where the system must complete triples in a KG without external retrieval. We evaluate this setting using the Language Model for Knowledge Base Construction (LM-KBC) benchmark, which provides datasets and metrics.

The LM-KBC challenge evaluates the ability of LLMs to construct or complete a knowledge base using parametric knowledge only [6]. The task provides head entity / relation pairs $\langle h, r \rangle$ from Wikidata and requires systems to output the complete set of tail entities t_i such that the triple $\langle h, r, t_i \rangle$ is in the graph. Relations vary in answer type and cardinality: the set of correct answers may be empty, a singleton, or contain many elements; some relations expect a numeric tail. Official scoring uses exact string matching for string entities and a 5% error margin on numerical entities, then computes macro-F1 score. The 2025 edition evaluates participants on the 6 relations listed hereafter with different challenges as described below and illustrated with an example. Previous editions were using other relations and triples but with the same objective and metrics.

- **countryLandBordersCountry**
 - *Number of pairs/split*: 67–68
 - *Challenge*: 0/1/many answers; empties possible
 - *Example (from the train set)*: Sierra Leone → Guinea, Liberia
- **personHasCityOfDeath**
 - *Pairs/split*: 100
 - *Challenge*: can be empty
 - *Example (train)*: Egbert Mulder → Groningen
- **hasCapacity**
 - *Pairs/split*: 100
 - *Challenge*: numeric object
 - *Example (train)*: Sungui Sports Complex in Incheon → 35000
- **hasArea**
 - *Pairs/split*: 100

- *Challenge*: numeric object
- *Example (train)*: Wellington Island → 5556
- **awardWonBy**
 - *Pairs/split*: 10
 - *Challenge*: potentially long lists (up to 224)
 - *Example (train)*: UEFA Team of the Year → Mesut Özil, Paolo Maldini, Luís Figo, Gerard Piqué, Cristiano Ronaldo, David Beckham, Franck Ribéry, Inter Milan, Lionel Messi
- **companyTradesAtStockExchange**
 - *Pairs/split*: 100
 - *Challenge*: can be empty
 - *Example (train)*: Farfetch → New York Stock Exchange

The 2025 edition imposes a fixed publicly available base model (Qwen3-8B [7]) and forbids external retrieval and model fine-tuning.

In this thesis, we focus on this LM-KBC dataset and we adopt closed-book Knowledge Base Construction (KBC) to characterize what can be retrieved from the model’s parametric memory. Accordingly, we avoid external retrieval and prioritize schema-aware prompt design and test-time scaling; later, we report exploratory Parameter-Efficient Fine-Tuning (PEFT) results whose goal is to standardize the task interface rather than inject new facts.

1.3 Research Objectives

Within the LM-KBC setting described above, we study how far one can go by designing better test-time procedures around a fixed base model and, more cautiously, by updating only a small number of parameters. On the inference side, we would like to impose relation-wise reasoning patterns and take advantage of the stochasticity of the generation process while respecting KG schema constraints. On the training side, we seek to understand whether parameter-efficient fine-tuning can improve retrieval of parametric knowledge in the data-scarce regime imposed by the dataset. This thesis approaches these challenges with the three following research questions:

1. Can we use generated synthetic data to guide the model towards a systematic reasoning that improves its knowledge retrieval ability?

2. How can test-time scaling methods such as self-consistency take advantage of the KG structure and schema?
3. Although fine-tuning is invasive, can PEFT methods help better retrieve parametric knowledge than prompting in data-scarce settings?

As the LM-KBC datasets don't contain valid reasoning paths, we build HumanCoT, a dataset that contains a few correct reasoning paths implementing a systematic way to address each relation and that is available in our repository ¹. This HumanCoT dataset is used in few-shot manner to generate SyntheticCoT, a larger dataset that contains reasoning paths that lead from train set subject entities / relation pairs to correct tail entities. It is also available in our repository ².

Reasoning paths from SyntheticCoT are used in few-shot prompting manner to generate predictions on the test and validation sets. We run multiple predictions for each input pair, and adapt the aggregation step from self-consistency in order to take advantage of the relation schema. The aggregation strategy for each relation is automatically tuned on the validation set, making it relation-wise self-consistency method we call ReWiSe and that we released ³. We participate and win the 2025 edition with this system. Then, for relations that expect a long list of object entities, we experiment with query reformulation, automatically separating the initial query into independent subqueries.

SyntheticCoT can also be used for training. We train PEFT models on this dataset to test the relevance on training as compared to prompting in this data-scarce setting.

1.4 Thesis Outline

Chapter 1 introduces the context and central concepts in LLM probing as well as KG construction challenges.

Chapter 2 reviews background on large language models, in-context learning, test-time scaling, parameter-efficient fine-tuning, knowledge graph construction, and knowledge probing methods relevant to knowledge base construction from language models.

Chapter 3 presents our methods for language model probing for knowledge base construction in closed-book settings. It first introduces our challenge-winning solution ReWiSe, relation-

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

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

³<https://github.com/Lama-West/ReWiSe>

wise self-consistency, then Relation Tuning, the PEFT adaptation of ReWiSe, and finally Sub-query, our improved strategy for relations that expect a large number of tail entities.

Chapter 4 describes experiments and reports experimental results on the LM-KBC 2025 dataset as well as on a dataset which is the result of all editions of the competition, from 2022 to 2025. It specifies evaluation protocols, metrics, and analyzes performance by relation. It also includes comparisons between inference-only systems and PEFT-based variants.

Chapter 5 summarizes findings, discusses limitations, and outlines directions for future work on reliable knowledge base construction from parametric knowledge.

CHAPTER 2 BACKGROUND AND LITERATURE REVIEW

2.1 Large Language Models

2.1.1 Pre-training and Parametric Memory

A PLM is a neural network trained for next-token (or masked-token) prediction over Internet-scale corpora. During this training, it acquires knowledge on a range of domains. This knowledge can be factual and it is stored in the model's parameters and early work on BERT [8] showed that the model could recall relational facts with competitive performance [2], proving that there is potential in harvesting knowledge from a PLM's parameters. Most of the recent PLM are based on the Transformer architecture [9] and the parameter count most often reaches the billions. However, the knowledge in an LLM is not stored in a usual database or knowledge graph. It is latent, distributed, and entangled across all of their parameters [10]. The model's knowledge is encoded in a diffuse manner that is not human-interpretable [11], making it harder to study the knowledge inside the model, or to use "language models as knowledge bases" [2]. Mechanistic interpretability attempts to identify and locate knowledge neurons or circuits that correspond to a specific fact but it is compute intensive and its application to studying broad knowledge are bleak.

Long-tail Knowledge and Distribution Effects. LLM tend to perform much better on frequent facts than on rare facts. The probability that a model knows a given fact appears to be highly correlated with how often that information was encountered in its training data: model accuracy in completing a triple with probing is strongly related to the number of training documents that mentioned the entities and relations [12]. Increasing the model size does improve the performance on long tail data but not sufficiently. More recent work introduced the PopQA benchmark focusing on Wikidata facts about less popular entities, and found that zero-shot LLM accuracy drops substantially for infrequent entities [13]. The authors argue for a Retrieval-Augmented Generation (RAG) architecture: augmenting inference by retrieving text or structured evidence from an external corpus or knowledge graph and conditioning the model on that context. RAG does prove better in the triple completion task but it not relevant when studying the knowledge stored in the model's parametric memory as it uses an external database.

2.1.2 In-context learning

In-context learning designates the ability of large language models to adapt their outputs to a task from examples and instructions embedded directly in the prompt, without gradient updates. This capability to use additional context to improve the output has been studied since GPT-3 with zero-shot, one-shot, and few-shot prompts spanning diverse tasks [14]. Subsequent instruction-tuning makes models more responsive and format-faithful to natural-language instructions and Q–A exemplars [15]. A practical question is how to design the prompt and how to select or arrange demonstrations so as to make the model reliably exhibit the intended behavior while minimizing hallucinations and spurious formatting, especially for knowledge-centric tasks. In-context learning also does not require parameter editing so it comes without the training cost, the need for a large training dataset and issues that arise from training explained in Section 2.1.4.

Prompting

Prompt wording strongly affects factual recall. Early probing work made this explicit on cloze statements such as "Dante was born in [MASK]" [2] by showing that rephrasing the template could yield largely different accuracies on the same fact, motivating template mining and paraphrasing to reduce prompt variance. LPAQA [3] introduced automatic template augmentation per relation, then aggregated answers across templates to improve recall.

Few-shot prompting

Few-shot prompting places a handful of task demonstrations (inputs and desired outputs) before the target query, reducing format ambiguity and often improving recall versus zero-shot [14]. One advantage of few-shot prompting is that it is an efficient way to impose a structure for the model’s generated text, which motivates our choice for this method.

Instruction-tuning

Instruction-tuning further increases adherence to natural-language instructions and target formats, making few-shot prompts more reliable in practice [15]. The difference between the terms Pre-trained Language Model and Large Language Model is that the prior has not been instruction-tuned whereas there is no such assumption for the latter. The instruction-tuning process is not meant to teach the model information and it is commonly considered that the model’s acquires its knowledge from the pre-training step. As most of the more recent LLMs, Qwen-3 [7] is instruction-tuned so it is designed to answer questions in a natural way.

2.1.3 Test-time scaling

Test-time scaling describes all the methods that increase inference-time compute to increase performance. These methods operate without changing parameters. Test-time scaling can be done either by lengthening a single generation (sequential) or by sampling multiple generations (parallel). We define the two core strategies here and instantiate them later.

Chain-of-Thought reasoning

CoT prompts elicit intermediate natural-language reasoning steps before the final answer. It is a sequential test-time scaling method that trades the compute cost of generating additional tokens and using a longer context length for improved reliability on many reasoning and factual tasks [16]. CoT often emerges in larger models and can be taught or scaffolded for smaller ones via carefully designed prompts or by specific training. Importantly, it can be done without training and thus fits our closed-book goal. Recently, some models were trained to improve their own CoT with reinforcement learning [17] which means that the model was trained to generate the reasoning path that helps it reach the best answer instead of training models to reason using human-based heuristics or trial and error. This kind of training is mostly done in reasoning or mathematical areas and not in parametric knowledge retrieval but it should be an interesting way to tackle LLM knowledge probing. It is not adapted to our context because it requires model training and because this training has to be conducted on a large model.

Self-Consistency

Self-consistency is a parallel test-time scaling method that samples multiple independent solutions to the same prompt and aggregates answers, using consensus as an implicit confidence signal [18]. It is purely test-time, complements CoT, and typically improves accuracy at the cost of more sampling.

2.1.4 Fine-tuning

Fine-tuning a model means that we continue its training on a targeted task. It often requires a large specific dataset. The motivation for fine-tuning can be to teach the model a specific task or facts. We are not interested in the latter as it doesn't align with the goal of studying the knowledge that is in the PLM: model fine-tuning consists in editing the model's parameters so it may alter the model's parametric memory. But we are interested in teaching the model a new task. Full model fine-tuning is expensive as it updates all parameters, and it comes with

issues that are listed below but PEFT freezes the backbone and learns a small set of additional parameters or modules. PEFT methods can be used in a data and compute-efficient way for task learning.

Issues with full model fine-tuning

Full model supervised fine-tuning is data and compute intensive as it requires training the whole model. Additionally, multiple issues may arise when training the model on a different objective such as catastrophic forgetting of the model's previous knowledge [19, 20]. Fine-tuning a model on data-intensive tasks can provoke frequency shock issues leading to under- or over-representation of entities having different frequencies in the pre-training and fine-tuning sets [21]. Fine-tuning on knowledge that is new and inconsistent with the pre-trained model's knowledge is slow and encourages hallucinations [22].

Prompt-based methods

The prompt is the input to a LLM. Prompts that are semantically similar may not produce similar answers [23]. This motivates the search for optimal prompts. A prompt is a sequence of tokens, or items of the model's vocabulary, and this vocabulary is discrete. Each token is associated to an embedding vector in a continuous space. Optimization in a continuous space can be achieved through gradient descent which is why methods try to optimize the prompt in the embedding space rather than in the text, or tokens, space.

Prompt Tuning [24] proposes to create new tokens, called "soft tokens" that are added to the model's vocabulary. These soft tokens are prepended to the prompt, and trained on the auto-regressive language-modeling task through gradient descent while the rest of the model is kept frozen. For knowledge probing, learned prompts act as relation- or task-specific keys that nudge the model to surface latent facts without changing them as the model is not edited.

Prefix Tuning [25] learns key-value prefixes injected into the self-attention of each layer, instead of only in the embedding layer, adding more trainable parameters and thus capacity to the tuning mechanism. Because the parameters that are added are not only in the embedding layer, they cannot be initialized with text and require more training data to be trained.

P-tuning [26] interleaves text tokens and soft (or "continuous") tokens. A second version of P-Tuning [23] interleaves those text and soft tokens across all layers as Prefix Tuning.

Adapter- and low-rank updates

Adapters. Classic adapters insert small bottleneck layers in transformer blocks and train only those, enabling task switching by swapping adapter sets and reducing forgetting [27]. Subsequent variants (e.g., AdapterFusion [28]) combine multiple adapters for transfer.

LoRA injects trainable low-rank matrices that act as rank- r updates to frozen weight matrices; only the low-rank factors are trained [29]. LoRA and its quantized version QLoRA [30] are now common baselines for efficient adaptation, with theoretical and empirical support that many task updates lie in low-dimensional subspaces.

This yields a family of PEFT methods. In data-scarce settings, methods with fewer trainable parameters tend to be more sample-efficient [31]; among those presented, Prompt Tuning trains the fewest parameters.

2.2 Knowledge Graphs

KGs provide a graph-based, uniform and machine-actionable abstraction of world knowledge in which entities are the nodes and relations are the edges entities structured by relations [1]. In this thesis, KGs play two complementary roles. First, they are the formalization of the target structure we wish to construct or complete from a LLM’s parametric memory. Second, the examples from the train and validation sets provide constraints and statistics over the schema and cardinality that can be exploited at inference time. A better understanding of the relations will improve the system’s reliability, as we will show with our relation-wise adaptation of self-consistency. This section reviews the classic KG completion and construction techniques, more recent LLM-based approaches, KG-aware language modeling, and evaluation issues relevant to our closed-book LM-KBC setting.

2.2.1 Formalization

A KG is a set of triples $\langle h, r, t \rangle$ with a head entity h , a relation r , and a tail entity t . Public, general-purpose KGs instantiate a rich schema: a formal and high-level structure or semantics that the graph follows, using classes, properties, qualifiers and references. They are maintained through a mixture of community curation and automated extraction and are, by their nature, incomplete: they are intended to accumulate knowledge over time. In the Semantic Web stack, this is formalized by the Open World Assumption (OWA), under which the absence of a triple in the graph does not entail that the fact is false [1]. By contrast, many link-prediction or KG completion protocols adopt Closed World Assumptions (CWA)

in practice—treating unobserved triples as negatives for training and/or evaluation—to enable negative sampling and rank-based scoring [32, 33]. This OWA–CWA tension motivates probing models on large, broad-coverage graphs.

Wikidata [34] provides concrete scale and diversity to anchor this thesis. As a large and general purpose initiative, it spans general knowledge. In the context of LM-KBC, this enables the benchmarking of models and probing methods on diverse knowledge by using a subset of Wikidata. The LM-KBC presents its data in the form of $\langle h, r, [t_i] \rangle$ triples where $[t_i]$ is the set of all tail entities such that given a query h, r , $\langle h, r, t_i \rangle$ is in Wikidata.

2.2.2 Automatic construction of Knowledge Graphs

From text and semi-structured data

Traditional KG construction pipelines break the problem into entity recognition, entity linking, relation extraction, coreference, and normalization, followed by schema alignment and quality control [1, 35]. Over two decades, each subtask has moved from feature-engineered methods to deep neural architectures trained with some supervision [36–38]. Contemporary surveys emphasize persistent challenges: coverage and long-tail entities, normalization across heterogeneous sources, and maintaining provenance and references at scale [1, 39, 40]. Despite significant progress, end-to-end high-precision construction in open domains remains costly because it depends on high-quality upstream data and human verification [41].

LM-KBC forbids use of external corpora and the knowledge must come from the model’s parametric memory [6]. Nevertheless, text-based KG construction methods can be relevant if applied on text that we generate with a LLM instead of relying on an external knowledge source.

From language models

We focus here on approaches that use LLMs to directly construct KGs. Despite the name of the LM-KBC competition, its goal is probing rather than construction: it evaluates how a model is queried under a fixed schema, not how a database is built. Constructing the graph itself is not the primary objective of this thesis, but the methods surveyed in this subsection would benefit from improved probing procedures, since they rely on similar query–response mechanisms to materialize a KG.

Because LLMs internalize extensive factual knowledge during pre-training, a growing line of work investigates using them as primary sources for KG construction [42–45]. This line of

research conceptualizes PLMs as "Open Knowledge Graphs". Their focus is to automatically extract broad knowledge from the models with minimal supervision, and store it in a classical database. Prompt-based crawling expands a graph around seed entities by iteratively asking an LLM for relational facts and consolidating them, with explicit attention to precision/recall trade-offs and halting criteria [43]. More recently, large-scale efforts have demonstrated that materializing LLM knowledge in a KG is possible at non-prohibitive cost, while surfacing open problems in canonicalization (entity/property naming), taxonomy induction, and provenance [46]: GPTKB [44] builds a KG with a hundred of million of triples, to be compared with Wikidata’s 1.65 billion triples, using recursive graph expansion and focusing on large-scale coverage, and entity-level canonicalization in an open-world setting. This line of work addresses the pipeline to build a KG from the parametric memory of a LLM but faces evaluation challenges. Follow-up work on GPTKB [45] also reports persistent hallucination and noise issues.

In this thesis, we focus on knowledge probing. As it is one of the steps of the KG construction pipeline used in methods such as GPTKB [45], advances in knowledge probing would benefit such initiatives. The resulting KG would be a better representation of the LLM knowledge. Probing is an easier task to evaluate than KG construction as we can fix a schema and evaluation benchmark and study how generation and aggregation procedures affect which triples are recovered.

2.2.3 Knowledge Graph Completion

Knowledge graph completion refers to predicting missing links in an existing graph. Methods fall into three broad families that differ in how they use structure and text.

Translational and bilinear embeddings

Early neural completion models embed entities and relations in a vector space and define a differentiable scoring function $f(h, r, t)$ trained on positive and corrupted triples. Translational models such as TransE [47] view relations as translations in latent space. In this paradigm, a triple is represented as a geometrical property in a latent space. Multiple variations moved away from the translation to use other geometrical properties to assimilate relations between entities in a KG (e.g., RotatE [48]). Bilinear models (e.g., DistMult, ComplEx) parameterize relations as matrices (or complex diagonal forms) acting on entity embeddings [49, 50]. Tensor-factorization variants such as TuckER generalize this view with core tensors [51]. These models scale and perform well on multi-relational benchmarks, but they rely purely on learning the graph structure and labels from many triples and do not leverage rich textual

semantics. Applications of this range of methods on LM-KBC are bleak as the context is reversed: they require a lot of structured training data, which LM-KBC does not provide, and they don't take advantage of unstructured knowledge such as the parametric knowledge that a model has. The LM-KBC challenge assumes semantic knowledge from LLMs but prevents the learning of the graph structure and entities: most entities appear once in the whole dataset and the train set is small.

Convolutional and GNN-based models

Architectures such as ConvE introduce convolution over reshaped embeddings to increase expressivity at moderate parameter cost [52]. Relational graph neural networks (e.g., R-GCN) and compositional variants (e.g., CompGCN) explicitly propagate information along typed edges and can incorporate local subgraph patterns [53, 54]. These models are effective in closed-world, transductive settings but inherit limitations from benchmark idiosyncrasies and negative sampling practices.

Text- and PLM-augmented KG completion

To bridge structure and semantics, text-aware approaches use entity/relation names and descriptions. KG-BERT casts triples as textual sequences scored by a pre-trained encoder [55]. KEPLER jointly optimizes language modeling and knowledge embedding objectives so that textual encoders produce entity representations aligned with KG structure [56]. These hybrids often improve zero- or few-shot generalization and inductive prediction on unseen entities but require careful negative sampling and are sensitive to description quality. PLM-augmented [55–57] methods which do not require external data at inference-time constitute the best performing closed-book KG completion systems but they require training on a large dataset. LM-KBC forbids training and does not provide a large dataset, so these methods are neither allowed nor relevant in this context.

2.2.4 Evaluation and Pitfalls

Standard completion evaluation typically assumes a closed world: any triple not observed in the KG is treated as false. Under this assumption, models are ranked by Mean Reciprocal Rank (MRR) and Hits@K [1, 58]. However, for open, incomplete KGs, this setting can mis-measure progress as stronger models may be penalized for predicting facts that are correct but missing from the current graph. Recent analyses argue for open-world evaluation and show metric degradation and inconsistency effects tied to focus-on-top ranking metrics [33].

Dataset design also matters. Inverse-relation leakage in early Freebase and WordNet splits inflated link-prediction scores: some triples (h, r, t) from the train set had their inverse relation (t, r^{-1}, h) in the test set so memorizing local patterns was boosting the performance although the desired objective is to learn generalizable relational structure. FB15k-237 and WN18RR were introduced to mitigate this by filtering out these kind of patterns, at the cost of lower overall performance and different inductive biases [52, 59–61]. For this thesis, these observations motivate macro-averaged metrics to prevent easy-gains from one simple query. It also motivates the choice of the triples in the LM-KBC benchmarks: underlying knowledge from a test-set triple cannot be inferred from the train set. In addition, as the challenge organizers argue, in order to create a system that builds a KGs, it is more relevant to have a disambiguation mechanism and to predict tail entities given a query rather than ranking them [6]: this protocol evaluates the binary choice of including a triple in a KG we would construct.

2.2.5 Knowledge Graphs and Language Models

This thesis is also part of a wider ongoing process to unify KG knowledge and LLM usability and capability to learn from large and unstructured data sources [62].

From KGs to LLMs

Knowledge-infused pre-training attempts to inject symbolic structure into PLMs. ERNIE modifies pre-training objectives to sensitize models to entities and relations [63]. KEPLER jointly learns textual and KG objectives so that descriptions become faithful proxies for entity embeddings [56]. Such methods are sought to improve downstream tasks requiring factual grounding but operate by editing parameters; they are therefore orthogonal to our closed-book objective of characterizing what a frozen model already knows but progress in this area could augment PLM’s parametric memory and therefore the performance of LLM knowledge probing.

From LLMs to KGs

Conversely, LLMs can populate or refine KGs at inference time. Beyond prompt-only extraction, RAG frameworks can augment KG queries with text retrieval. Graph-aware variants such as GraphRAG [64] uses graph topology to structure retrieval and summarization. In the setting of the 2025 edition of LM-KBC we study, such retrieval is disallowed.

2.3 LLM Knowledge Probing

We study how to elicit, structure, and assess the knowledge residing in a frozen LLM’s parameters. In our setting, models cannot be fine-tuned and cannot query external corpora; probing must therefore rely on prompt design and test-time scaling only, with evaluation grounded in KG schema and cardinality.

2.3.1 Prompt-based probing

Early work established that pretrained language models can recall many relational facts without task-specific training when queried with cloze templates such as "Barack Obama was born in [MASK]" [2]. The task of predicting the [MASK] token is designed to be similar to the pre-training task. Following on this work, a branch of Natural Language Processing (NLP) studies how to probe PLM and extract their knowledge [3, 65–67]. However, factual recall is highly sensitive to the phrasing of the prompt: semantically equivalent templates yield different results, which motivates template mining and paraphrase ensembling per relation (LPAQA) [3]. In our LM-KBC context, this implies that robust extraction requires careful, relation-specific prompt design and mechanisms that reduce prompt variance. We adopt both [68].

Within the prompting methods that improve LLM probing, few-shot prompting reduces format ambiguity by placing demonstration exemplar pairs before the target query [14]. Instruction-tuning further increases adherence to natural-language instructions and desired output formats, making few-shot prompts more reliable in practice [15]. Because LM-KBC fixes the base model to an instruction-tuned LLM, we lean on few-shot exemplars to stabilize parsing and steer outputs toward schema-compliant strings and numbers.

2.3.2 Test-time scaling for knowledge probing

Chain-of-thought prompting

CoT prompting elicits intermediate natural-language reasoning before the final answer and often improves reliability by trading more tokens for better deliberation [16, 69]. While strong CoT emerges in larger models, it can be distilled and taught to smaller models as Fine-tune-CoT [70] and STaR [71] have shown. The imposed Qwen-3 [7] is distilled from a CoT-specialized model, so it was post-trained to do this reasoning. In our no-training ReWiSe approach, we try to distill the reasoning capacity and performance from a larger model to the imposed smaller model with prompting. Beyond LM-KBC’s rules which prevent model

training, in a second step, we experiment with PEFT to perform the model adaptation to the systematic reasoning in an efficient way.

Self-consistency

Self-consistency [18] samples multiple independent solutions to the same prompt and aggregates them. Consensus acts as a reliability signal with the intuition that the correct answer can be supported by multiple independent reasoning paths. The first version of self-consistency [18] gets rid of the reasoning path and uses majority voting among the answers as the way to determine consensus. This makes sense for their application task which is multiple-choice or true/false questions. While it is possible to compute majority voting for our LLM knowledge probing predictions, we adapt it such that it acts at the entity-level and takes advantage of the KG structure. Some of the more recent versions of self-consistency [72–74] use the reasoning path in the consensus process and generalize to a wider range of contexts, but no version was designed to take advantage of the KG structure.

Universal Self-Consistency replaces answer extraction and majority voting with an LLM-based selector that chooses the most semantically consistent candidate among multiple samples, extending self-consistency beyond short-form answers to code, long-form, and open-ended outputs [72].

Recursive schemes expand and reconcile chains before aggregation. Recursive Self-Aggregation repeatedly reformulates and fuses reasoning paths, showing that multi-round consolidation can outperform single-shot voting [73].

Other recent variants weight votes by estimated confidence, reducing the number of samples needed for reliable decisions [74].

Query reformulation and decomposition

Queries that have many correct answers are particularly challenging as a single query often misses coverage [75, 76]. Two complementary strategies are common. First, paraphrase ensembling reduces prompt variance by probing the same query with multiple templates [3]. Second, decomposition-based prompting splits a broad query into targeted sub-queries that partition the answer space. Representative approaches include least-to-most prompting, which solves a sequence of simpler subproblems [77], and decomposed prompting, which delegates sub-tasks to modular prompts and can recursively subdivide inputs [78]. Related techniques such as Self-Ask formulate explicit question decomposition steps [79]. For KBC, a natural way to mitigate coverage issues on many-tail relations is to decompose a broad query

into sub-queries (for instance by time, geography, or entity type), probe each sub-query separately, and aggregate their answers. Similar decomposition ideas are used in complex KG-based question-answering, where multi-clause questions are answered by combining answer sets from simpler sub-questions [80].

Distractors and counterfactual alternatives

Reliability can be stress-tested by confronting the model with plausible but incorrect alternatives. Self-ensemble checks compare independently sampled answers for agreement to flag likely hallucinations without retrieval [81]. Distractor-based evaluations inject irrelevant or misleading content to measure robustness; recent controlled studies show that even strong models are sensitive to such perturbations in their reasoning paths [82]. Beyond binary correctness, fine-grained factuality metrics (e.g., FActScore) assess whether atomic claims are supported by evidence [83]. For probing parametric knowledge under LM-KBC, distractors can be used to calibrate consensus thresholds and to validate that agreement is not achieved by matching spurious patterns.

2.4 Summary and open challenges

This chapter reviewed how KGs are built and completed, how LLMs internalize and expose factual knowledge, how prompt design and test-time scaling improve a frozen model’s performance, and how PEFT can be used to improve the model itself. Existing KG completion methods either assume large, closed-world graphs or rely on LLMs trained on substantial supervision, which is incompatible with the small, open-world, closed-book LM-KBC setting. KG construction from LLMs shows that large graphs can be materialized from parametric knowledge, but it still faces hallucinations. It typically uses probing procedures; prompt-based probing and self-consistency demonstrate that many facts can be elicited without training, yet they remain sensitive to prompt choice, do not enforce relation-specific reasoning patterns, and ignore KG schema and relation cardinality. Finally, PEFT offers a way to specialize models with limited parameters, but its behavior for knowledge probing in data-scarce regimes is still underexplored.

In the remainder of this thesis, we focus on a subset of these gaps that fit the LM-KBC constraints. Chapter 3 introduces our methods for closed-book probing on a Wikidata based KG: building a synthetic dataset of relation-wise reasoning patterns, adapting self-consistency to operate at the entity and relation level, decomposing high-cardinality queries into semantically-aware sub-queries, and exploring a parameter-efficient fine-tuning to test

whether light training can improve parametric knowledge retrieval beyond prompting alone in data-scarce settings.

CHAPTER 3 LANGUAGE MODEL PROBING METHODS FOR KNOWLEDGE BASE CONSTRUCTION

This chapter formalizes our approach to LLM probing to extract knowledge from the parametric memory. We first describe the LM-KBC task dataset and its evaluation protocol, then present our competition submission method, ReWiSe, which combines reasoning traces generation with relation-wise entity aggregation. We then extend ReWiSe with a query partition mechanism for high-cardinality queries. Finally, we report an out-of-competition fine-tuning study, Relation Tuning, whose goal is to improve the model’s understanding of queried relations using parameter-efficient methods instead of prompting only.

3.1 The LM-KBC competition

The Language Model for Knowledge Base Construction competition is an annual event that challenges participants to develop methods for extracting factual knowledge from LLMs. The competition focuses on the tail entities prediction task, evaluating the ability of LLM to generate the set of object entities given a subject entity and a relation. The task is designed to assess the open-world cardinality of relations, allowing for zero, one, or many valid object entities.

Formally, in a KG, given a head entity h and a relation r , the goal is to predict the set of tail entities $T = \{t_1, t_2, \dots, t_n\}$ such that each (h, r, t_i) is in the KG.

As the goal of the competition is to evaluate LLM’s factual knowledge, entities appear once in the dataset, preventing the use of entity learning or retrieval-based methods that rely on entity repetition. The competition has been held annually since 2022, with different sets of relations, rules and evaluation criteria. Evaluation criteria are detailed further in section 3.1.3.

3.1.1 Datasets

The relations used for evaluation have changed across editions but the task remains the same. We mainly use Dataset2025, the dataset of the 2025 edition of LM-KBC. In addition, we build a unified dataset which is the union of all LM-KBC editions: 2022 to 2025. We make the union split by split, clean it to remove duplicates and leaks from the train to the validation or test split. As the answers of the test set are private, we evaluate on the validation set. We make sure to use an independent run on the validation set (not a run that was used to

set hyperparameters) for evaluation. This unified dataset will enable us to have an idea of the performance of the methods we develop on a larger dataset and on other relations. We make the unified dataset available as well as the script to build it. Table 3.1 shows statistics on relations from this unified dataset. Similar statistics on the 2025 dataset were presented in Section 1.2.

All test datasets are kept private by the challenge organization. Thus we cannot conduct evaluation on the test dataset other than our official prediction. Therefore, most of our results will be run on the validation set. We acknowledge that this is not ideal as the validation set is also used to tune our aggregation mechanism, but we cannot evaluate against the test set.

3.1.2 Competition rules

Across editions, the core task is unchanged: for each given subject entity and relation, systems must predict the complete set of valid object entities, allowing zero, one, or many objects, and some relations expect numeric objects. Subjects appear only once across the splits to prevent memorization. Here is a quick summary of the evolutions of LM-KBC rules.

2022 Edition

In the 2022 edition, participants were required to output materialized, disambiguated entities, i.e., Wikidata identifiers that can be inferred from the Wikidata disambiguation tool from a string. There were two tracks, one restricting to the model BERT [8] one open on the model used.

2023 Edition

In 2023, there were two tracks again but with different constraints. The small model track limited submissions to models under one billion parameters and disallowed additional context, while the open track had no constraint on the model or the use of external context. As in 2022, systems had to output disambiguated entities as Wikidata identifiers, but some of the relations were expecting numerical outputs.

2024 Edition

In 2024, the edition used a single track with a 10 billion parameters cap. Outputs continued to be disambiguated entities as Wikidata identifiers except for numerical outputs, and the use of external context was allowed.

Relation	Train	Val	Test	Total
BandHasMember	100	100	100	300
ChemicalCompoundElement	33	97	99	229
CityLocatedAtRiver	100	100	100	300
CompanyHasParentOrganisation	154	144	146	444
CountryBordersCountry	48	104	97	249
CountryHasOfficialLanguage	35	99	97	231
CountryHasStates	46	46	46	138
FootballerPlaysPosition	100	100	100	300
PersonCauseOfDeath	183	149	150	482
PersonHasAutobiography	100	100	100	300
PersonHasEmployer	190	150	147	487
PersonHasNoblePrize	100	100	100	300
PersonHasNumberOfChildren	100	100	100	300
PersonHasProfession	200	150	150	500
PersonHasSpouse	100	100	100	300
PersonPlaysInstrument	193	150	147	490
PersonSpeaksLanguage	199	150	150	499
RiverBasinsCountry	109	143	135	387
StateBordersState	99	100	100	299
StateSharesBorderState	100	50	50	200
awardWonBy	10	10	10	30
companyTradesAtStockExchange	100	100	100	300
countryLandBordersCountry	67	68	67	202
hasArea	100	100	100	300
hasCapacity	100	100	100	300
personHasCityOfDeath	123	212	213	548
seriesHasNumberOfEpisodes	198	199	200	597
Total	2987	3021	3004	9012

Table 3.1 Unified dataset: per-relation subject counts across splits

2025 Edition

In 2025, fine-tuning was disallowed and the use of external corpora for retrieval (no RAG) was forbidden to focus on the model’s parametric knowledge rather than on its capacity to handle retrieved knowledge. Unlike previous editions, systems were to output strings rather than Wikidata identifiers. This means that "USA" or "United States" instead of "United States of America" won’t be considered correct answers anymore although the Wikidata disambiguation tool would have recognized them as being the same entity in the graph. Therefore, the 2025 evaluation is stricter for string-based entities. It does allow for a 5% error margin so it is less strict on numerical entities.

3.1.3 Evaluation and metrics

Precision, recall, F1-score

The LM-KBC challenge uses precision, recall and F1-score metrics at the entity level. Given an prediction object entity list $P := [p_1 \dots p_k]$ and a ground truth object entity list $G := [g_1 \dots g_m]$, precision and recall are defined as follows:

$$\text{Precision} = \begin{cases} 1 & \text{if } k = 0 \\ \frac{|P \cap G|}{k} & \text{otherwise} \end{cases}$$

$$\text{Recall} = \begin{cases} 1 & \text{if } m = 0 \\ \frac{|P \cap G|}{m} & \text{otherwise} \end{cases}$$

Then, the F1-score is computed as follows:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Note: If one list is empty but not the other, the F1-score is 0.

Macro F1-score

The LM-KBC challenge ranks performances using the Macro-F1 score, which is the double average of F1-scores across all relations and their respective object entity lists:

$$\text{Macro-F1} = \frac{1}{R} \sum_{r=1}^R \left(\frac{1}{N_r} \sum_{i=1}^{N_r} \text{F1}_{r,i} \right)$$

where R is the total number of relations, N_r is the number of object entity lists for relation r , and $\text{F1}_{r,i}$ is the F1-score for the i -th entity list of relation r .

Evaluation with Wikidata disambiguation

Previous editions of the LM-KBC challenge (2022-2024) used the Wikidata entity disambiguation tool:

- **Numerical entities:** Required exact match with no margin for error
- **Non-numerical entities:** Used Wikidata disambiguation to resolve entity aliases.

For example, the tool would give the same entity id Q30 for USA, `United States` and `United States of America`. The F1-score is computed at the entity id level.

Evaluation with exact string matching

The 2025 edition of the LM-KBC challenge updated the evaluation criteria:

- **String entities:** Evaluation uses string matching, disallowing to use of aliases in entity names. This approach is stricter but it makes sense as the 2025 edition of the challenge forbids Internet access, which is necessary in order to use the Wikidata disambiguation tool;
- **Numerical entities:** A 5% margin of error is accepted for numerical predictions. This is necessary because relations such as `hasArea` expect float outputs.

This 2025 edition metric is the default metric that we'll use throughout the thesis. We will use it as it is for the 2025 edition dataset. For the unified dataset, which is the union of datasets from 2022 to 2025, we use a case insensitive version of the 2025 metric as there is no convention for the case that holds throughout the datasets.

3.1.4 Baseline and limitations

Each dataset provides a relation-to-prompt mapping. Each relation is paired with a natural-language template into which the subject entity is inserted, so the (relation, subject) pair is represented directly as a question. Since 2023, templates are phrased as questions suitable for generative probing. Table 3.2 lists the templates used for each relation of the 2025 baseline. Using these templates is optional.

The competition baseline uses five-shot prompting as shown in the example right below. For each test instance, it samples five question–answer examples at random from the training set that share the same relation. These examples will be used as few-shot prompting examples: the prompt presents them with the question template paired with its gold answer list. The test query is then formed by inserting the test subject into the same template, and the model is expected to return the corresponding answer list. The five-shot context both illustrates the expected output format (comma-separated list or "None") and clarifies the relation through concrete examples. The system instruction shown in the example below is part of the baseline prompt format. The baseline uses question prompting: using a question which expects the tail entities as answer. All question templates that are used are presented in the Table 3.2.

Relation	Prompt template
hasArea	What is the area of {subject_entity} in square kilometers?
hasCapacity	What is the total capacity of {subject_entity}?
countryLandBordersCountry	Which countries share a land border with {subject_entity}?
personHasCityOfDeath	In which city did {subject_entity} die?
awardWonBy	Who won the {subject_entity}?
companyTradesAtStockExchange	Where do shares of {subject_entity} trade?

Table 3.2 Dataset 2025 baseline: relation-to-question templates

Official baseline — personHasCityOfDeath

System: 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 (question template instantiated from a train set subject entity): In which city did Jan Turski die?

Assistant (gold answer associated, from train set): Warsaw

User (few-shot examples 2, from train set): In which city did Basina Kloos die?

Assistant (gold answer of few-shot examples 2, from train set): None

... 5 few-shot examples in total ...

User (using an entity from the test or validation set): In which city did Jeroen Brouwers die?

————— generated —————

Assistant (used as prediction): Maastricht

For the 2025 edition, the baseline does not enable the model’s chain-of-thought behavior.

3.2 Relation-Wise Self-consistency

We propose a new approach called Relation-Wise Self-consistency (ReWiSe) [68] 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. ReWiSe is our solution for LM-KBC. It therefore has to be compliant with the challenge’s rules.

3.2.1 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 repository¹.

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 [84]. Triples included in HumanCoT

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

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.

HumanCoT `countryLandBordersCountry` example: Kenya

User: Which countries share a land border with Kenya?

Assistant (HumanCoT example): <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.

Examples: Subjects: Ricky Nelson (dead) and Konrad Rufus Müller (alive)

HumanCoT personHasCityOfDeath examples: Ricky Nelson (dead) and Konrad Rufus Müller (alive)

User: Where did Ricky Nelson die?

Assistant (HumanCoT example): <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

User: Where did Konrad Rufus Müller die?

Assistant (HumanCoT example): <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

awardWonBy. This relation is challenging as the number of ground truth outputs can be large (224 in the train set). 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.

HumanCoT awardWonBy examples: Fields Medal, 1990 recipients

User: Who won the Fields Medal?

Assistant (HumanCoT example, 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 reasoning strategy that would help the model, 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 CoT used for these three relations below:

hasCapacity. For this relation, we provide 3 examples of human reasoning paths.

HumanCoT hasCapacity example: Gaddafi Stadium

User: What is the total capacity of Gaddafi Stadium?

Assistant (HumanCoT example): <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.

HumanCoT hasArea example: Heimaey

User: What is the area of Heimaey in square kilometers?

Assistant (HumanCoT example): <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.

HumanCoT companyTradesAtStockExchange example: Novadios

User: Where do shares of Novadios trade?

Assistant (HumanCoT example): <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

3.2.2 SyntheticCoT Dataset

Synthetic generation

The SyntheticCoT dataset is generated for triples from the training set using few-shot prompting with examples from HumanCoT. The goal is to have a larger set of correct reasoning traces although in practice we can only ensure we have reasoning traces that lead to

a correct answer. 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. The experimental details of the SyntheticCoT construction are presented in Table 3.3.

Table 3.3 Relation Statistics for Llama 70B Model. The SyntheticCoT dataset consists of both correct and incomplete outputs, totaling 2878 examples.

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

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. An example below presents one correct, one incorrect and one incomplete sample. The correct and the incomplete ones are kept while the incorrect is filtered out.

SyntheticCoT Filtering — countryLandBordersCountry

In this example, we illustrate how samples from SYNTHETICCoT are filtered during dataset construction. We show three generations for the same subject (*Bolivia*), representing respectively a correct (kept), incomplete (kept), and incorrect (discarded) case. Filtering relies on comparison with the gold set of neighboring countries.

Case: Correct (kept)

... Elapsed system message and few shot examples ...

User: Which countries share a land border with Bolivia?

Assistant (the final answer is correct, same set as ground truth): <think> Bolivia is a landlocked country in western-central South America. It is surrounded by five countries: to the north and east by Brazil, to the southeast by Paraguay, to the south by Argentina, to the west by Chile, and to the northwest by Peru. </think> Brazil, Paraguay, Argentina, Chile, Peru

Case: Incomplete (kept)

... Elapsed system message and few shot examples ...

User: Which countries share a land border with Bolivia?

Assistant (the final answer is incomplete, included in the ground truth but misses Peru): <think> Bolivia is a landlocked country located in central South America. To the north, it shares a border with Brazil. To the east, it also borders Brazil. Moving south, Bolivia shares a border with Paraguay. Then, continuing west, it shares borders with Argentina and Chile. </think> Argentina, Brazil, Chile, Paraguay

Case: Incorrect (discarded)

... Elapsed system message and few shot examples ...

User: Which countries share a land border with Bolivia?

Assistant (the final answer is incorrect, as it contains Uruguay): <think> Bolivia is a landlocked country in central South America. It borders several countries, including Brazil to the north and east, Peru to the northwest, Chile to the southwest, Paraguay and Argentina to the south, and Uruguay to the East. </think> Argentina, Brazil, Chile, Paraguay, Peru, Uruguay

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 3.3 for per-relation details. The dataset is available in our repository².

3.2.3 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. Few-shot prompting provides the model with examples of what is expected. When using few-shot prompting with SyntheticCoT examples, the model should copy the reasoning strategy introduced in HumanCoT and copied to SyntheticCoT. This approach also helps establish the expected format, which is essential for processing the generated answers.

For the generation step, we use 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 $(CoT_i, a_i)_{1 \leq i \leq n_c}$.

An example of the prompt used at inference time is available below.

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

ReWiSe Inference Prompt — personHasCityOfDeath

System: 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 (from SyntheticCoT): In which city did Milan Lasica die?

Assistant: `<think>` Milan Lasica was a Slovak actor, writer, and dramatist. He died in 2021 in Bratislava. `</think>` Bratislava

... 5 few-shot examples in total, all from SyntheticCoT with the same relation ...

User (entity we query on): In which city did Jeroen Brouwers die?

————— generated —————

Assistant: `<think>` Jeroen Brouwers was a Dutch writer and essayist. He passed away in 2022 in Maastricht, a city where he spent much of his later life. `</think>` Maastricht

3.2.4 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. The goal of self-consistency is to increase the performance at the cost of extra compute. When we generate n_c independent answers, the inference cost is also scaled by n_c . Aggregation strategies is not compute-intensive but it helps to make the most of the generated answers. Standard self-consistency aggregates answers (a_i) using majority voting. In our setup, however, each answer is a list of object entities $a_i = [t_{i,1}, \dots, t_{i,j}]$, 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.

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 θ 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 θ_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].

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.

Example

ReWiSe Inference — `personHasCityOfDeath` (Subject: Nicolae Martinescu)

In this example, we present two of the twenty reasoning paths and predictions produced by ReWiSe with CoT. We also provide the aggregated table of all predictions. For this relation (`personHasCityOfDeath`), the aggregation method is majority voting, considering "None" as a valid candidate.

2 out of 20 samples from the inference process

Sample 1 — correct

System: 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.

... 5-shot examples from SyntheticCoT ...

User: In which city did Nicolae Martinescu die?

Assistant: `<think>` Okay, let's try to figure out where Nicolae Martinescu died. First, I need to recall who Nicolae Martinescu was. He was a Romanian historian and philosopher, right? I think he was active in the 20th century. Now, I need to find out the city of his death. I remember that he was associated with the University of Bucharest. Maybe he died there? But I'm not sure. Let me think. I should check some key points. If he was a Romanian figure, his death might be recorded in Romanian sources. I'm not certain, but I think he might have died in Bucharest. Wait, but maybe I should verify. Alternatively, could he have died in another city? I don't recall any specific information about him passing away in a different city. Given that he was a scholar in Romania and his work was mainly in Romania, it's plausible that he died in Bucharest. But I'm not 100% sure. However, based on available information, Bucharest seems the most likely answer. So, I'll go with Bucharest. `</think>` Bucharest

Sample 2 — incorrect

System: 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.

... 5-shot examples from SyntheticCoT ...

User: In which city did Nicolae Martinescu die?

Assistant: `<think>` Okay, the user is asking in which city Nicolae Martinescu died. Let me think. First, I need to recall who Nicolae Martinescu was. He was a Romanian writer and critic, right? I remember he was associated with the Romanian literary scene in the 19th century. Now, I need to find out the city of his death. Wait, I think he died in Exile. But Exile isn't a city. Maybe he died in a specific city while in exile. Let me try to remember. I recall that he spent time in Paris, France. Was he in Paris when he died? Or maybe another city? I'm a bit fuzzy here. Alternatively, perhaps he died in Bucharest, his home country. But I'm not sure. Let me try to verify. Wait, some sources mention that Nicolae Martinescu died in Paris in 1874. But I need to confirm. Another possibility: maybe he died in another European city. Alternatively, maybe he died in a place related to his exile. I think the correct answer is Paris. But I'm not 100% certain. Let me check my memory again. Yes, I think he died in Paris. So the answer should be Paris. `</think>` Paris

Majority voting over 20 predictions Here are all of the answers that were made using the 20 generations. CoTs were discarded for agglomeration.

Candidate	Count (n=20)
Bucharest	6
None	5
Paris	4
Târgu Jiu	1
Rome	1
Cluj-Napoca	1
Timișoara	1
Buiucani	1

Final aggregated prediction: Bucharest (correct). In this example, the correct answer is recovered through aggregation even though it only appeared in 40% of individual generations, illustrating the benefit of answer-level self-consistency.

3.3 Sub-queries for high-cardinality relations

3.3.1 Motivation and formalization

Motivation

Some relations admit a large object entities set for a given subject entity, which makes a single query prone to omissions. For example, the query `Nobel Prize in Physiology or Medicine, awardWonBy`, which is in the training dataset, has 227 correct outputs. Predicting them all at once is a challenging task. We empirically find that the model consistently predicts the same few object entities although it has some knowledge about other correct entities. For example, using Qwen3-8B, the query "Who are the female recipients of the Nobel Prize in Physiology or Medicine?" yields Gerty Cori (although a correct answer) zero times out of five, while the broader query "Who won the Nobel Prize in Physiology or Medicine?" yields Gerty Cori five out of five. This motivates an approach that decomposes a broad query into several independent sub-queries that partition the answer space, then aggregates their results at the entity level. This is a "Divide and Conquer" approach inspired from the work from an other competitors team in the 2025 edition of LM-KBC [85]. We implement it afterwards but we generalize their approach as described at the end of this section.

Problem formulation

Given a head entity and a relation in a KG, the goal is to return the set of tail entities. For high-cardinality one-to-many relations, we generate a fixed number of sub-queries for the base query. Each sub-query targets a coherent subregion of the answer space such as time, geography, category, demographic, or any semantic subregion of the initial query's answer space. Each sub-queries is asked independently to the LLM and aggregated with entity-level self-consistency per sub-queries as if they were different queries for ReWiSe. We then predict every entity that was in one of the sub-query's aggregated prediction. The compute cost of this method scales linearly with the number of sub-queries.

3.3.2 Implementation

This section describes our implementation of Sub-queries. It targets relations where the true object list set is long and naturally divisible into subsets. In LM-KBC 2025, it concerns the relation `awardWonBy`. Although a team of competitors used a similar strategy in the 2025 edition of LM-KBC [85], they use a fix and `awardWonBy`-specific division of the query dividing into decades and recipient nationalities. We ask the model to generate variants of the initial

question template, each variant represents a sub-query, making the approach applicable on queries involving another relation that would expect a large number of object entities.

We first present the 3 steps of our methodology and then illustrate it with an example that follows the 3 steps.

Sub-query generation

As we use question variants for the Sub-query abstraction, we propose a concise instruction that operates on the base query rendered as a question for the model interface, while the task abstraction remains a query over entities.

The instructions encourage the model to partition the answer space along high-level dimensions such as time, geography, category or demographic groups, and to avoid single-entity or yes or no questions. The ten sub-queries are stored and reused for the inference step. Below is an example of these instructions.

Sub-query Generation Prompt — awardWonBy (Subject: *Turing Award*)

User: You are given a broad question with many answers: Who won the Turing award?

Goal: Write 10 sub-questions that partition the answer space so that, together, they cover most of the original answers. Overlap is allowed, but each sub-question should still yield multiple answers.

Method (reason internally; output only the questions):

- Pick 2-4 high-level partitioning dimensions (e.g., time, geography, category, demographic).
- For each, choose broad categories (no singletons).
- Turn categories into independent sub-questions without changing meaning or adding assumptions.

Constraints:

- No yes/no or single-entity questions. Avoid duplicates.
- Keep wording tight and aligned with the base question.
- Questions will be asked independently, so avoid dependencies and make sure they can stand alone. For example, make sure to repeat the context or relevant information for each question.

Output exactly 10 lines:

Q1: ...

Q2: ...

...

Q10: ...

Inference per sub-query

In order to have a coherent prompting scheme, we formulate HumanCoT exemplars with a question variant instead of the usual template, and with the answer associated to the sub-query defined by the template. Using this version of HumanCoT, we generate SyntheticCoT with question variations and their generated answer, although the filtering process has to be done on the full list as we can't verify that an entity is a correct answer to the specific sub-query.

For each sub-query, we apply the same inference stack as in ReWiSe. We use few-shot prompting with these new SyntheticCoT exemplars for the target relation, generate n_c independent answers, then perform entity-level aggregation within the sub-query using the

relation-specific rule chosen on validation.

Merging across sub-query

Let \hat{T}_v be the aggregated entity set for sub-query v . The final prediction for the base query is the union over sub-queries, with exact-string de-duplication:

$$\hat{T} = \bigcup_v \hat{T}_v$$

This union increases recall while keeping precision under control through the per-sub-query aggregation step that already filters low-consensus entities.

3.3.3 Example

Sub-queries — awardWonBy (Subject: *Sakharov Prize*)

In this example, we illustrate the full Sub-queries pipeline on the subject entity *Sakharov Prize*. Ten Sub-queries are generated by Qwen3-8B during the sub-query generation step, each phrased differently to probe distinct subsets of possible answers. These sub-queries are then asked independently in inference, each 20 times with CoT reasoning enabled. After generation, we apply self-consistency with a threshold of $\theta = 0.1$, keeping candidates that appear at least twice in the 20 samples per sub-query. The union of all filtered per-sub-query sets forms the final prediction.

sub-query generation:

1. Which individuals won the Sakharov Prize from European Union member states?
2. Which individuals won the Sakharov Prize from non-European countries?
3. Which individuals won the Sakharov Prize in the 2000s?
4. Which individuals won the Sakharov Prize in the 2010s?
5. Which individuals won the Sakharov Prize in the 2020s?
6. Which individuals won the Sakharov Prize for human rights activism?
7. Which individuals won the Sakharov Prize for freedom of expression?
8. Which individuals won the Sakharov Prize as part of groups or organizations?
9. Which women won the Sakharov Prize?

10. Which men won the Sakharov Prize?

Inference per sub-query: Few-shot prompt (5 examples from SyntheticCoT)

For illustration, the sub-query "Which individuals won the Sakharov Prize for human rights activism?" is asked $n_c = 20$ times. Here is the example of the generation for one of these 20 times. After the standard system message, we give 5 few-shot examples from SyntheticCoT

System: 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 (from SyntheticCoT with sub-queries): Who won the Fulbright Prize for their work in the field of humanities?

Assistant (from SyntheticCoT with sub-queries): `<think>` The Fulbright Prize is awarded to recognize individuals who have made significant contributions to the field of humanities. Some notable winners include Nelson Mandela and Jimmy Carter. Craigan Urdaneta is not a winner, but Václav Havel received the award for his efforts in promoting peace and understanding through the humanities. `</think>` Nelson Mandela, Jimmy Carter, Václav Havel

... 5 few-shot examples from SyntheticCoT with sub-queries in total ...

User (sub-queries we make inference on): Which individuals won the Sakharov Prize for human rights activism?

————— generated —————

Assistant: `<think>` The Sakharov Prize is awarded by the European Parliament to individuals or groups who have dedicated their lives to the defense of human rights and freedom of thought. Past laureates include well-known activists such as Ales Bialiatski and Liu Xiaobo, as well as figures like Aung San Suu Kyi, Andrei Sakharov himself, and Denis Mukwege. Others, including Nadia Murad and Maria Ressa, have also been recognized for their advocacy. These are some of the main human rights recipients. `</think>` Ales Bialiatski, Liu Xiaobo, Aung San Suu Kyi, Andrei Sakharov, Denis Mukwege, Nadia Murad, Maria Ressa

Merging across sub-queries

For the agglomeration threshold $\theta = 0.1$, every prediction that appears twice as answers to this sub-query is kept. In this case the list of answers that are kept for this sub-query is Liu Xiaobo, Ales Bialiatski, Aung San Suu Kyi, Andrei Sakharov, Václav Havel, Lech Wałęsa, Denis Mukwege, Nadia Murad, Adolfo Pérez Esquivel, Maria Ressa, Elie Wiesel, Nelson Mandela, Shirin Ebadi, Dalai Lama, Adam Michnik, Luis Moreno Ocampo, Mehdi Karrouchi and Mohamed Choukri.

These retained entities form the answer set for this single sub-query. We then use predicted entities for each of the sub-queries to form our final prediction. In this case, there were 31 entities in the prediction.

3.4 Relation Tuning

The 2025 edition of LM-KBC does not allow model fine-tuning but it is a useful tool that can be leveraged in the context of KG construction from LLM. This section describes Relation Tuning, a study intended for knowledge graph construction beyond the LM-KBC 2025 rules.

3.4.1 Motivation

The objective is to improve the model’s understanding of the queried relations. The training dataset contains 477 samples, which is little for model fine-tuning but it is under-used by the few-shot prompting strategies: only 5 of these are used by the baseline. PEFT methods typically require fewer training samples than full model fine-tuning and the fewer trainable parameters there are the more data-efficient the training [31]. The train set and SyntheticCoT are both quite small even for PEFT but it is worth trying them.

This study aims to adjust the model’s utilization of its existing knowledge rather than to introduce new facts. The aim of the training here is not to give the model more knowledge that can be later retrieved but rather to teach it reasoning schemes, output format and some understanding about the relations such as the typical number of outputs or some details that might not be as clear from the sheer prompt template.

3.4.2 Training

Prompt Tuning

Prompt Tuning [24] is a PEFT method. A few new tokens called "soft" or "continuous" are added to the model’s vocabulary and only these new parameters are updated when training the model while the rest of the model is kept frozen. By adding k tokens, we add and

train $k \times d_{embed}$ parameters. For Qwen3-8B which has $d_{embed} = 4096$ and using a typical number of soft tokens $k = 10$, there are 40960 trainable parameters which represents very few parameters relatively to the 8 billions in the model.

The trainable soft prompt is prepended to the textual query and its parameters are updated during the training while the rest of the model remains frozen. We tried two configurations, one in which there are k soft tokens per relation and one with k total soft tokens. The difference is that in the prior case, soft tokens are trained on fewer data samples and they are specialized in each relation while in the latter case, each soft token is trained on the whole dataset and the specialization is on the task rather than on a specific relation. Performance difference was not significant and we decided in favor of the unified (the latter) configuration as we think that there is some transfer learning happening because the probing task is similar from a relation to another one.

Training data

Training can be performed with CoT on SyntheticCoT or without CoT on the competition train set. A model trained on the train set will generate the list of object entities immediately, with no reasoning trace, while a model trained on SyntheticCoT will first generate a CoT then the list of answers, following the same format. This enables us to perform a comparison between CoT and no CoT configurations. While this can help us understand the implications of CoT, we can't compare the training runs directly as the training sets don't represent the same data as described in Table 3.4.

Training objective

The model is trained with next-token prediction on the target text that follows the query. Given a prompt that is identical to the one it would receive in zero-shot prompting, it is trained to generate the reasoning path and the answer in CoT mode, and only the answer in the no-CoT mode. Both cases are illustrated by examples in Section 4.1.5.

Initialization

Prompt Tuning adds new trainable tokens in the model's vocabulary. Instead of initializing at random, it is possible to initialize them using copies from existing tokens from the model's vocabulary, making the soft tokens training initialize from a better than random point. In our experiments, using the unified model, we start with the text "Answer the question".

Dataset	Competition Train set	SyntheticCoT
Number of samples	477	2878
Unique subject–relation pairs	477	344
CoT	No	Yes
Answer quality	Correct	Correct + incomplete

Table 3.4 Comparison between Relation Tuning training data sources

3.4.3 Inference

At inference time, Relation Tuning uses the same query rendering as ReWiSe. We sample n_c independent generations per query and apply the relation-wise entity-level aggregation described in Section 3.2.4. The only inference difference between ReWiSe and Relation Tuning is that instead of using few-shot prompting, the trained soft tokens are prepended to the prompt. In that sense, the k-long soft prompt acts as a compressed alternative of the few-shot prompt.

3.5 Overview of the presented methods

Taken together, the methods presented in this chapter define a probing framework for LM-KBC. We start from the official few-shot baseline and extend it with human-authored and synthetic CoTs, then we introduce a relation-wise, schema-aware version of self-consistency. These components are combined in ReWiSe. For high-cardinality relations, Sub-queries further decomposes broad queries into targeted sub-queries with semantic coherence and whose predictions are merged at the entity level. Finally, Relation Tuning explores a complementary, parameter-efficient adaptation of the base model that aims to better align its parametric knowledge with the LM-KBC interface. The next chapter evaluates these components empirically across LM-KBC editions and discusses their respective impact to reliable knowledge extraction from LLMs.

CHAPTER 4 EXPERIMENTS AND DISCUSSION

In this chapter, we describe the experiments that we conducted to evaluate and compare the methods presented above. We then show the results that we obtain, and discuss these results.

4.1 Experimental setup

4.1.1 Datasets

The ground truth answers from test sets of all LM-KBC editions are private. Therefore, we cannot evaluate on them directly. We do have results per relation for our official submissions, but we don't have access to test set evaluation for other methods: neither for Relation Tuning and the Question Variation methods not for any run on the unified dataset. For this reason, we evaluate on the validation set, which is also used for the strategy calibration in the relation-wise self-consistency aggregation. We use independent runs for evaluation and calibration, but we recognize that running the evaluation on the validation set could lead to an overestimation of the method. Results from ReWiSe method are presented on both the validation and the test set and show coherent results across both sets.

4.1.2 Models and hyperparameters

- Models used
 - Qwen3-8B (Inference, training, variation generation)
 - Llama 3.3-70B Instruct (SyntheticCoT generation)
- Inference hyperparameters
 - Temperature is always 1.0
 - Few-shot: $k = 5$ as in the baseline (unless specified otherwise)
- Relation-wise self-consistency hyperparameters
 - Annex A shows an ablation study in which we study the effect of the self-consistency parameter n_c from 1 to 100. We empirically find that a good value for the self-consistency level is $n_c = 20$ so we fix it for all experiments that use self-consistency. $n_c = 1$ is equivalent to no self-consistency

- For one-to-many relations, we test thresholds θ from 0.0 to 1.0 with 0.05 increments. This provides the same granularity as the self-consistency with $n_c = 20$
- For numerical relations, we compare mean, median and majority vote as described in Section 3.2.4

4.1.3 System prompts

We use the following system prompts in our experiments:

- **CoT system prompt:** 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.
- **Inference system prompt when using question prompts:** 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”.
- **Inference system prompt when using relation prompts:** Given a subject entity name and a relation name, your task is to provide the list of object entity names without any other context. If there are multiple answers, separate them with a comma. If there are no answers, type “None”.

4.1.4 SyntheticCoT construction, experimental details

The SyntheticCoT dataset is built as described in Section 3.2.2. We use HumanCoT examples as few-shot prompts for each relation, which serve as a guide to synthesize new reasoning paths. For the generation process, we use Llama 3.3 70B Instruct as our generator model to distill reasoning capabilities into the synthetic examples. We apply a 2-shot prompting approach where examples are randomly selected from HumanCoT for each triple in the training set. To increase diversity and coverage, we repeat this process 10 times per training example – but using different HumanCoT examples. This process is meant to leverage both the stochasticity of the generation process and the variety of few-shot selections from HumanCoT.

Prompt for SyntheticCoT Generation — personHasCityOfDeath

System: 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, including Angela Merkel. I don't have information suggesting that he has passed away. `</think>` None

User (question on an entity we don't have in HumanCoT): In which city did Jeroen Brouwers die?

————— generated —————

Assistant (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 another nearby city, but Maastricht is the best-known location associated with him. `</think>` Maastricht

After generation, we apply a filtering process that retain only outputs that are either exactly correct or incomplete (strict subsets of ground truth for set-valued relations) when evaluated under the 2025 metric. All other outputs, including those with incorrect predictions, are discarded. This filtering strategy is particularly important for relations with high cardinality, where generating the complete exact list is challenging.

As shown in the Table 3.3 presented in Section 3.2.2, the generation process has yield 4,590 reasoning paths in total, with 2,794 (60.9%) classified as correct and 84 as incomplete. The relation `countryLandBordersCountry` achieves the highest correctness rate at 91.7%, suggesting that this relation is easier to complete from the model's parametric memory. Conversely, no completely correct examples were generated for `awardWonBy`, highlighting the difficulty of high-cardinality relations. We rely on incomplete answers that provide partial lists of award winners. The full ground truth object entities list may be recovered through

our self-consistency aggregation mechanism.

4.1.5 Relation Tuning experimental setup

We implement Relation Tuning as described in Section 3.4, using prompt tuning as our PEFT method. We use 10 dedicated soft tokens. These tokens are initialized from existing tokens corresponding to the relation template, which provides a semantically meaningful starting point rather than random initialization. Each relation receives its own separate set of prompt tuning parameters, effectively creating as many fine-tuned model variants as relation.

The training process uses the standard next-token prediction loss (cross-entropy) with the training target depending on the configuration: either generating only the answer or generating both the chain-of-thought reasoning path and the answer. We train for a maximum of 10 epochs and evaluate on validation set performance.

For the training data, we leverage the LM-KBC train split, with two distinct approaches:

- training directly on answer generation as in the competition baseline
- using SyntheticCoT examples to train the model to generate both reasoning paths and answers.

The prompt templates used for these two approaches are shown below:

Relation Tuning — No CoT Training Example (Subject: Jeroen Brouwers)

The model is trained on the natural language generation tasks on what is after the "training" separator.

System: 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 from the train set): In which city did Jeroen Brouwers die?

———— training ————

Assistant: Maastricht

Relation Tuning — CoT Training Example (Subject: Jeroen Brouwers)

The model is trained on the natural language generation tasks on what is after the "training" separator.

System: 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 from SyntheticCoT): In which city did Jeroen Brouwers die?

———— training ————

Assistant: `<think>` Jeroen Emile Karel Brouwers was a famous civil servant of the Flemish community. It is said that he was shot in Maastricht. `</think>` Maastricht

4.2 Experiments

This section presents the experiments that we conduct to tune ReWiSe and Relation Tuning, compare them and understand their behavior. For each experiment, we specify the objective, the protocol and the metrics that we report and then we give the results.

4.2.1 Optimal aggregation strategy per relation

Objective

The first experiment determines, for each relation, the best aggregation strategy among those described in Section 3.2.4. This strategy is then reused in all subsequent experiments for that relation and inference configuration.

Protocol

We run an inference pass on the validation set with $n_c = 20$ and test all aggregation strategies on the resulting n_c predictions.

For one-to-many relations `countryLandBordersCountry`, `companyTradesAtStockExchange`, and `awardWonBy`, we compare thresholds θ ranging from 0.0 to 1.0 in increments of 0.05. Each threshold determines the minimum fraction of the $n_c = 20$ independent predictions in which an entity must appear to be included in the final aggregated answer.

For numerical relations `hasArea` and `hasCapacity`, we compare three aggregation methods: majority voting, mean and median. For the relation `personHasCityOfDeath`, only majority voting is used.

This experiment uses CoT prompting with 5 random SyntheticCoT examples as few-shot

samples to ensure consistent reasoning patterns. The exact same procedure is repeated using the train set directly instead of SyntheticCoT in order to later compare CoT and non-CoT methods. Whenever we change the inference process (Relation Tuning, question variations, unified dataset), we rerun this selection step, as the optimal strategy may differ.

Metrics and reported quantities

Performance is evaluated on the validation set using Macro-F1 scores. For each relation and inference configuration, we select the aggregation strategy that maximizes Macro-F1. The chosen strategies are summarized in a per-relation table:

- for each relation: selected threshold θ (one-to-many) or aggregation rule (numerical and singletons),
- Macro-F1 obtained on the validation set with this choice.

This table is used as a reference for all subsequent experiments and will be designated as Table 4.1.

Results

This experiment selects, for each relation on Dataset2025, the aggregation rule used in relation-wise self-consistency (Section 3.2.4). Figure 4.1 shows the validation set results for the CoT run and Figure 4.2 the no-CoT run, for each relation and across all tested strategies. Table 4.1 summarizes the chosen strategy per relation for both the CoT and non-CoT regimes with $n_c = 20$.

For one-to-many relations, Table 4.1 reports the consistency threshold θ_r that maximizes Macro-F1 on the validation set. For numerical relations, it indicates whether majority vote, mean or median over n_c generations is selected. For `personHasCityOfDeath`, which is single-valued, the aggregation always reduces to majority voting over entities (including the `None` token).

Overall, the selected strategies confirm the intuition that aggregation needs to be specialized by relation as the optimal strategy depends a lot on the relation. According to Figures 4.1 and 4.2, there is no one-size-fits-all strategy. One-to-many relations `awardWonBy` and `companyTradesAtStockExchange` favor relatively low thresholds, showing the need to retain entities that appear in only a low fraction of the samples, while `countryLandBordersCountry` uses a higher threshold. For numerical relations, mean is the best choice for `hasCapacity`

Relation	Relation type	CoT strategy	No-CoT strategy
awardWonBy	one-to-many	$\theta = 0.1$	$\theta = 0.1$
companyTradesAtStockExchange	one-to-many	$\theta = 0.35$	$\theta = 0.25$
countryLandBordersCountry	one-to-many	$\theta = 0.5$	$\theta = 0.8$
hasArea	numerical	Median	Median
hasCapacity	numerical	Majority Vote	Mean
personHasCityOfDeath	single-valued	Majority Vote	Majority Vote

Table 4.1 Per-relation aggregation strategy on Dataset2025 selected on the validation set for the CoT and no-CoT regimes with $n_c = 20$.

while it is a poor one for **hasArea**. These choices remain fixed for all subsequent experiments on Dataset2025.

4.2.2 ReWiSe ablations: Chain-of-Thought and self-consistency

Objective

The objective of this experiment is to disentangle the contributions of CoT prompting and self-consistency in ReWiSe. We want to quantify how much each component improves performance and whether they interact in a complementary way.

Protocol

We work on Dataset2025 and evaluate on the validation split, using the default inference setup described in Section 4.1 (Qwen3-8B, temperature 1.0, five-shot prompting unless specified otherwise).

We vary two factors:

- CoT prompting: disabled vs enabled.
- Self-consistency level: $n_c = 1$ (no self-consistency) vs $n_c = 20$.

This yields four configurations.

When CoT is disabled, we use the inference system prompt without reasoning tags from Section 4.1 and draw few-shot examples from the competition train set (direct question–answer pairs). When CoT is enabled, we use the CoT system prompt from Section 4.1 and draw few-shot examples from SyntheticCoT, as described in Section 4.1.4.

For each query and each configuration, we sample $k = 5$ few-shot examples from the same relation. When $n_c = 1$, we keep the single generated answer list. When $n_c = 20$, we generate

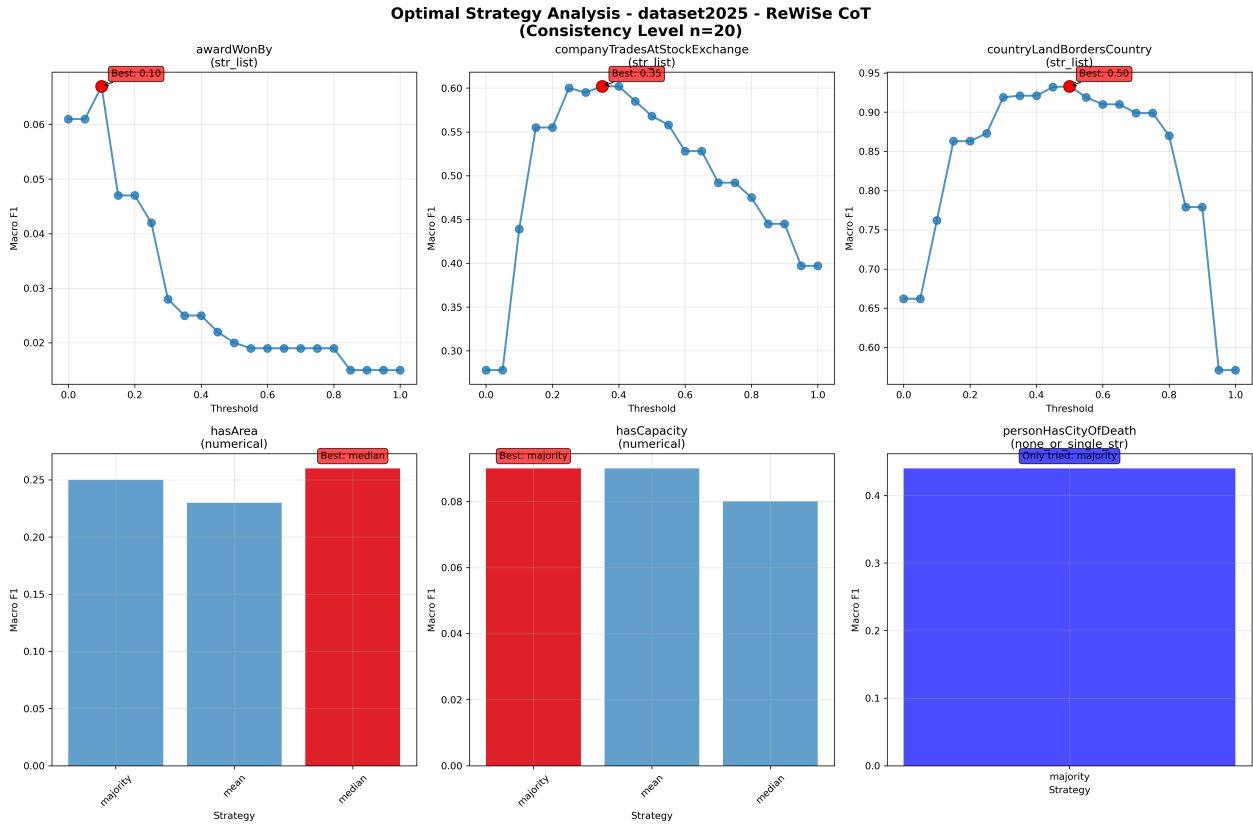


Figure 4.1 Relation-wise comparison of aggregation strategies using 20 independent predictions on the validation set **with CoT**. In red, the best strategy on the validation set. This strategy is chosen for the prediction.

twenty independent answer lists per query and aggregate them at the entity level using the per-relation strategies selected in Section 4.2.1.

Metrics and reported quantities

We evaluate each configuration on the validation set of Dataset2025 using Macro-F1 under the 2025 metric. We report:

- global Macro-F1 for each of the four configurations,
- per-relation Macro-F1 to see which relations benefit most from CoT, from self-consistency, or from both.

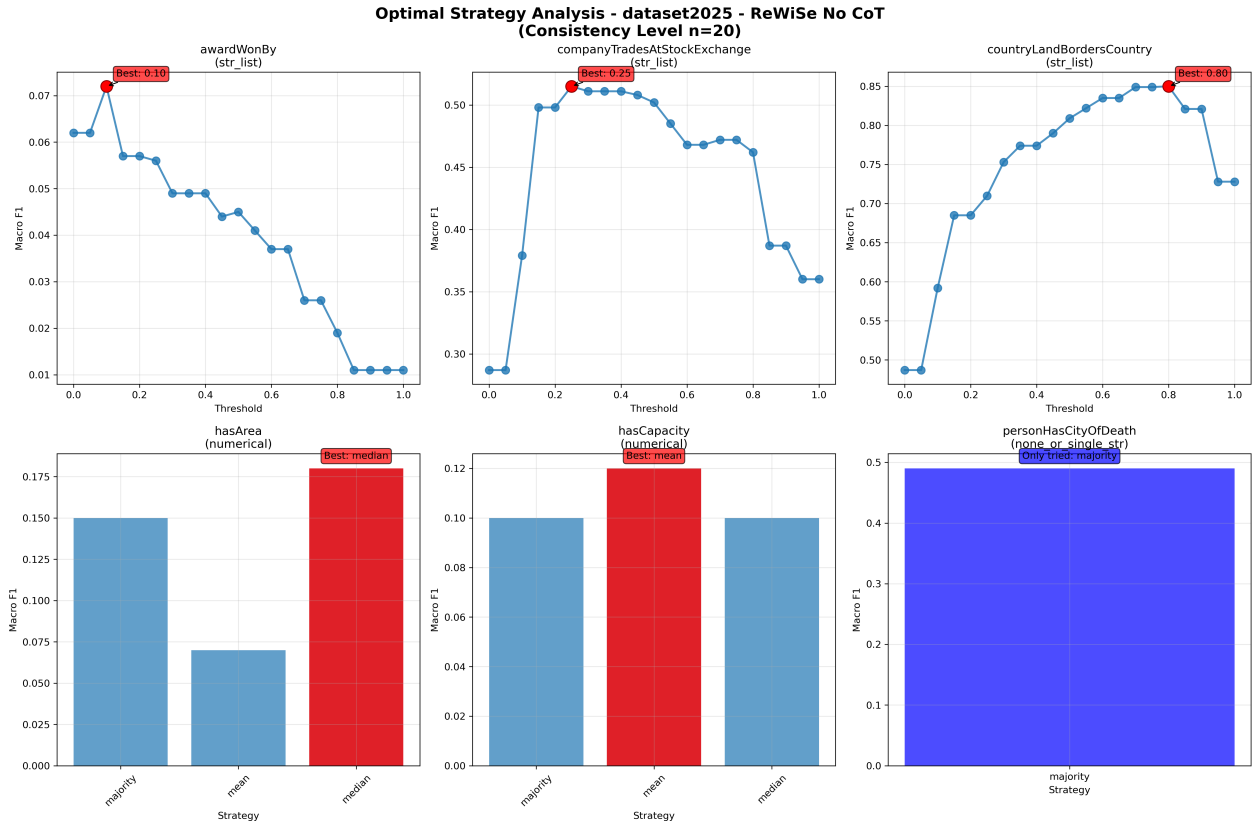


Figure 4.2 Relation-wise comparison of aggregation strategies using 20 independent predictions on the validation set **without** CoT. In red, the best strategy on the validation set. This strategy is chosen for the prediction.

Results

This experiment studies the effect of CoT prompting and self-consistency on Dataset2025, using Qwen3-8B and the aggregation strategies from Table 4.1.

Validation set. Table 4.2 reports Macro precision, recall and F1 on the validation set for all combinations of CoT usage (yes/no) and self-consistency level ($n_c \in \{1, 20\}$).

Two patterns stand out:

- Self-consistency is the main driver of improvement. Without CoT, Macro-F1 increases from 0.331 at $n_c = 1$ to 0.378 at $n_c = 20$. With CoT, it increases from 0.367 to 0.417.
- CoT brings an additional gain once self-consistency is enabled, but its impact is relation-dependent. It is clearly beneficial for `companyTradesAtStockExchange`, `hasArea` and `countryLandBordersCountry` and , while the effect is smaller or mixed for numerical

Table 4.2 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 CoT 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. 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
companyTradesAtStockExchange	No	1	0.610	0.555	0.426
companyTradesAtStockExchange	No	20	0.621	0.651	0.491
companyTradesAtStockExchange	Yes	1	0.755	0.533	0.463
companyTradesAtStockExchange	Yes	20	0.780	0.673	0.597
countryLandBordersCountry	No	1	0.729	0.900	0.767
countryLandBordersCountry	No	20	0.764	0.923	0.794
countryLandBordersCountry	Yes	1	0.868	0.918	0.858
countryLandBordersCountry	Yes	20	0.919	0.926	0.911
awardWonBy	No	1	0.705	0.027	0.036
awardWonBy	No	20	0.632	0.069	0.075
awardWonBy	Yes	1	0.345	0.010	0.018
awardWonBy	Yes	20	0.182	0.055	0.073
hasArea	No	1	0.240	0.150	0.150
hasArea	No	20	0.200	0.200	0.200
hasArea	Yes	1	0.290	0.260	0.260
hasArea	Yes	20	0.260	0.260	0.260
hasCapacity	No	1	0.270	0.070	0.070
hasCapacity	No	20	0.180	0.090	0.090
hasCapacity	Yes	1	0.520	0.060	0.060
hasCapacity	Yes	20	0.120	0.080	0.080
personHasCityOfDeath	No	1	0.800	0.500	0.410
personHasCityOfDeath	No	20	0.890	0.500	0.480
personHasCityOfDeath	Yes	1	0.605	0.540	0.387
personHasCityOfDeath	Yes	20	0.790	0.520	0.430
All Relations	No	1	0.520	0.395	0.331
All Relations	No	20	0.518	0.434	0.378
All Relations	Yes	1	0.585	0.422	0.367
All Relations	Yes	20	0.543	0.454	0.417

relations `hasCapacity` and for `personHasCityOfDeath`. For `awardWonBy`, all configurations remain difficult. This is the reason why we later develop the Sub-queries.

Overall, the best validation configuration is CoT and self-consistency with $n_c = 20$, which we adopt as the main ReWiSe setting for Dataset2025.

Test set. Table 4.3 compares the official LM-KBC 2025 baseline (no CoT, $n_c = 1$) with our best ReWiSe configuration (CoT, $n_c = 20$) on the hidden test set.

Table 4.3 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. In this case, the baseline is the official run from the competition organizers.

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

ReWiSe significantly outperforms the baseline on 5 out of 6 relations, with improvements of up to 48 points for `personHasCityOfDeath`. Only on `awardWonBy` does the baseline perform better, which is consistent with the high-cardinality difficulty observed on the validation set. Averaged over all relations, ReWiSe reaches a Macro-F1 of 44.4%, more than doubling the baseline score of 21.2%.

Taken together, validation and test results show a consistent results for ReWiSe proving that the analysis made on the validation set are relevant when no result is available on the test set. Adding self-consistency on top of single predictions improves performance in all regimes, and adding CoT on top of self-consistency yields the best overall scores, with the exception of the high-cardinality relation `awardWonBy`. Proposing a better method for this specific case is the object of our next experiment described in Section 4.2.3.

Competitors

We describe here the methods of the LM-KBC competitors. Then, Table 4.4 gives the final leaderboard with all participants and the baseline.

He and Razniewski. He and Razniewski [85] propose a pipeline that combines a self-RAG with a divide-and-conquer strategy. Their approach first produces intermediate, structured context intended to support answer generation under the closed-book setting, and then enforces strict output constraints to reduce formatting errors. For the high-cardinality relation `awardWonBy`, they decompose the task into multiple targeted and constrained subproblems to broaden coverage, and then consolidate candidates using a rule-based validation heuristics.

Soft Thinking. Creo et al. [86] introduce *Soft Thinking*, a parameter-efficient method that inserts trainable soft prompts directly into the reasoning segment of the model output. Instead of relying on explicit natural-language rationales, their method learns relation-specific latent reasoning cues that guide the model during generation while keeping the base model frozen. As opposed to our Relation Tuning, a separate set of soft tokens is used for each relation. This design aims to shape and optimize the internal reasoning trajectory with minimal additional parameters, and to improve robustness without extensive manual prompt engineering of reasoning path design.

Sam. Sam [87] focuses on a lightweight, inference-oriented approach centered on prompt engineering and decoding choices. The system explores structured prompts and generation settings aimed at improving answer formatting and reducing variance, while also evaluating a quantized (4-bit) configuration to lower computational cost.

Clay et al. Clay et al. [88] propose a multi-stage generate-and-filter pipeline where the model first generates candidate answers and then applies an additional judging step to select the most credible candidates. Their method uses the LLM both as a generator and as an evaluator that scores or verifies candidates, complemented with post-processing to handle extraction and formatting. This setup aims to improve precision by rejecting low-confidence outputs, at the cost of additional inference passes performed sequentially in a pipeline.

4.2.3 Sub-queries for `awardWonBy`

Objective

The relation `awardWonBy` admits a large set of object entities for most subject entities. A single query often leads the model to repeat the same few entities and omit many valid ones. The objective of this experiment is to assess whether decomposing such queries into a fixed set of model-generated sub-queries, as described in Section 3.3, improves recall while keeping precision under control when combined with relation-wise self-consistency.

Rank	Team	Macro-F1	Paper
1	Albert-Roulhac and Zouaq (ours)	44.4	[68]
2	He and Razniewski	40.5	[85]
3	Creo et al.	39.8	[86]
4	Sam	24.1	[87]
5	Clay et al.	21.6	[88]
6	Baseline	21.2	–

Table 4.4 LM-KBC 2025 final leaderboard (test set Macro-F1).

Protocol

We restrict this experiment to the relation `awardWonBy` on Dataset2025 and evaluate on the validation split. We follow the three-step procedure of Section 3.3.

Baselines and comparison. On `awardWonBy`, we evaluate six configurations. For each of CoT and no CoT, we first use single prediction ($n_c = 1$), which matches the competition baseline without CoT and its few-shot CoT counterpart; we then add self-consistency with $n_c = 20$ (ReWiSe and its no-CoT ablation); and finally we apply the Sub-queries mechanism on top of the same CoT/no-CoT regimes. All runs use Qwen3-8B and 5-shot prompting.

Few-shot prompts for `awardWonBy` (Subject: *Sakharov Prize*)

As the few-shot prompting scheme is the main difference between these experiments, we present here the four ways we present few-shot examples on the same subject *Sakharov Prize*.

Question using template with CoT (used in Few-Shot + ReWiSe, CoT runs).

User: Who won the Sakharov Prize?

Assistant (from SyntheticCoT): <think> The Fulbright Prize for International Understanding is awarded to recognize individuals who have made extraordinary contributions to international understanding and diplomacy. Past winners include Jimmy Carter, Nelson Mandela, and Bill Clinton. </think> Jimmy Carter, Nelson Mandela, Bill Clinton

Question using template without CoT (used in Few-Shot (Baseline) + ReWiSe, no-CoT runs).

User: Who won the Sakharov Prize?

Assistant (62 elements ground truth list from the train set): Salima Ghezali, Zacarias Kamwenho, Alexei Navalny, Reporters Without Borders, Alexander Dubček, ...

Sub-query with CoT (used in Sub-queries, CoT runs).

User: Who won the Fulbright Prize for their work in the field of humanities?

Assistant (from the Sub-query SyntheticCoT): <think> The Fulbright Prize is awarded to recognize individuals who have made significant contributions to the field of humanities. Some notable winners include Nelson Mandela, Jimmy Carter, and Craigan Urdaneta is not a winner, but rather Václav Havel received the award for his efforts in promoting peace and understanding through the humanities. </think> Nelson Mandela, Jimmy Carter, Václav Havel

Sub-query without CoT (used in Sub-queries, no-CoT runs).

User: Who won the Fulbright Prize for their work in the field of humanities?

Assistant (answer only from the Sub-query SyntheticCoT): Nelson Mandela, Jimmy Carter, Václav Havel

Metrics and reported quantities

We evaluate both methods on the validation split of Dataset2025 using Macro-F1 on `awardWonBy`. In addition to Macro-F1, we report precision and recall to quantify the trade-off introduced by sub-queries. We also present one detailed example showing:

- a set of generated question variants (for Sub-queries) for a given award,
- the entities recovered by each sub-query,
- the final union compared to the single-query baseline.

Results

For `awardWonBy`, we evaluate the Sub-query strategy from Section 3.3, which decomposes a broad query into several model-generated sub-questions and aggregates their predictions.

On the Dataset2025 validation split, we compare:

- the reproduction of the baseline and its CoT adaptation. Those methods use few-shot prompting respectively on the train set and on SyntheticCoT
- ReWiSe with and without CoT
- a sub-query-based configuration where each subject is queried through multiple sub-queries, aggregated with self-consistency at the sub-query level and a union over sub-queries at the entity level. This is also done with and without CoT. For the experiment that uses no CoT, we use examples from SyntheticCoT as they have the sub-query and the matching generated answers, but we don't use the CoT and use only the answer.

The corresponding Macro precision, recall and F1 scores are reported in Table 4.5.

Table 4.5 Performance on `awardWonBy` relation (validation set) comparing baseline, ReWiSe, and Sub-queries approaches. All experiments use Qwen3-8B with 5-shot prompting. Baseline uses $n_c = 1$ (single generation). ReWiSe uses $n_c = 20$ with self-consistency on identical questions. Sub-queries uses $n_c = 20$ with diverse question formulations generated by Qwen3-8B, aggregating predictions across different phrasings of the same query.

CoT	n_c	Method	Macro-P	Macro-R	Macro-F1
CoT	1	Few-Shot	34.5	1.0	1.8
CoT	20	ReWiSe	18.2	5.5	7.3
CoT	20	Sub-queries	14.1	11.8	11.0
No	1	Few-Shot (Baseline)	70.5	2.7	3.6
No	20	ReWiSe	63.2	6.9	7.5
No	20	Sub-queries	8.7	10.0	7.6

Table 4.5 shows that, even in the best configuration, Macro-F1 on `awardWonBy` remains low: the best score, obtained by Sub-queries with CoT, is 11.0. Moving from single prediction to self-consistency systematically helps in both regimes (from 1.8 to 7.3 with CoT, and from 3.6 to 7.5 without CoT). This is only adding the relation-wise self-consistency, the few-shot exemplars are presented in the same way. Adding Sub-queries on top of ReWiSe with CoT brings a clear improvement: Macro-F1 increases from 7.3 to 11.0, driven by a doubling of recall ($5.5 \rightarrow 11.8$) at the cost of a drop in precision ($18.2 \rightarrow 14.1$). Without CoT, however, Sub-queries make the precision fall from 63.2 to 8.7 while recall only increases from 6.9 to 10.0, leaving Macro-F1 essentially unchanged ($7.5 \rightarrow 7.6$).

This behaviour can be related to the different few-shot signals used by the methods. In the non-CoT regime, Few-Shot baseline and ReWiSe use exemplars where the answer list is very long. Essentially, the model is prompted to answer very long lists and by doing so it gets

high precision but very low recall, just making a very long list of names. The answers to Sub-queries are targeted and therefore the answers are shorter lists that make sense semantically as answers are more similar, since they are answers to the same sub-query.

In the CoT regime, the answer length is closer. Targeted questions divide the answer space and force the model to answer with elements that are not the obvious ones thus getting a larger coverage as shown by the recall increase between ReWiSe and Sub-queries. This gain is obtained at some computational price, as we produce n_c answers for each of the 10 sub-queries instead of n_c answers in total as in ReWiSe, but as shown in A, adding more compute by simply raising n_c was not sufficient to raise the F1 score when going beyond 20.

4.2.4 Relation Tuning

Objective

The objective of this experiment is to evaluate whether Relation Tuning, as defined in Section 3.4, can improve tail entity prediction on Dataset2025 compared to purely prompt-based ReWiSe in a data-scarce regime. We focus on identifying relations that benefit from training and on comparing configurations with and without CoT supervision.

Protocol

We work on Dataset2025 and evaluate on the validation split. The base model is Qwen3-8B. We use the unified prompt tuning configuration introduced in Section 3.4: a single soft prompt of length $k = 10$ shared across all relations, prepended to the textual query. Only the soft prompt parameters are updated; the rest of the model remains frozen.

We consider two training regimes, which reuse the training targets and prompt templates defined in the Relation Tuning experimental setup in Section 4.1:

- Training on the competition train set, with targets consisting only of the answer list (no CoT).
- Training on SyntheticCoT, with targets consisting of a reasoning trace between `<think>` and `</think>` tags followed by the answer list.

In both cases, the training objective is next-token prediction on the target text. We train for up to ten epochs and monitor Macro-F1 on the validation set after each epoch. We keep the checkpoint that achieves the best validation Macro-F1 for each regime.

At inference time, Relation Tuning uses the same query rendering as ReWiSe: the subject is inserted into the relation template or question prompt, and the trained soft prompt is prepended. We then generate answers with Qwen3-8B using the default inference settings from Section 4.1. For each query, we sample $n_c = 20$ independent generations and aggregate them at the entity level using the per-relation strategies selected in Section 4.2.1. No explicit few-shot examples are provided, since the soft prompt is expected to encode task information that would otherwise be supplied by examples.

Baselines, metrics and reported quantities

We compare:

- ReWiSe with CoT, five-shot prompting on SyntheticCoT and $n_c = 20$,
- Relation Tuning trained on the competition train set,
- Relation Tuning trained on SyntheticCoT.

All methods are evaluated on the validation split of Dataset2025 using Macro-F1 under the 2025 metric. We report:

- global Macro-F1 for each method,
- per-relation Macro-F1 to identify relations that benefit from Relation Tuning,
- where relevant, qualitative examples illustrating differences between prompt-based and trained behavior.

Results

We now compare prompt-based ReWiSe with Relation Tuning on Dataset2025. Relation Tuning models are trained for 10 epochs and evaluated with and without few-shot prompting at inference time.

Table 4.6 reports Macro-F1 for three regimes:

- ReWiSe with 5-shot prompting (ReWiSe 5-shot),
- Relation Tuning with 0-shot prompting (RT 0-shot),
- Same Relation Tuning model but with 5-shot prompting at inference time (RT 5-shot).

Table 4.6 Per-relation Macro-F1 comparison on dataset2025 validation set between ReWiSe and Relation Tuning with different prompting strategies. All experiments use Qwen3-8B with $n_c = 20$ (self-consistency). ReWiSe uses the base model with 5-shot prompting. RT 0-shot uses the trained model (10 virtual tokens, 10 epochs) with zero-shot inference. RT 5-shot combines the trained model with 5-shot prompting at inference time. Best result for each relation is in **bold** and second best is in *italic*.

Relation	Method	No CoT	CoT
companyTradesAtStockExchange	ReWiSe 5-shot	49.1	59.7
	RT 0-shot	45.1	36.0
	RT 5-shot	40.3	<i>51.0</i>
countryLandBordersCountry	ReWiSe 5-shot	79.4	91.1
	RT 0-shot	80.5	92.4
	RT 5-shot	79.7	<i>92.0</i>
hasArea	ReWiSe 5-shot	20.0	<i>26.0</i>
	RT 0-shot	7.0	13.0
	RT 5-shot	12.0	27.0
hasCapacity	ReWiSe 5-shot	<i>9.0</i>	8.0
	RT 0-shot	2.0	0.0
	RT 5-shot	6.0	11.0
personHasCityOfDeath	ReWiSe 5-shot	48.0	43.0
	RT 0-shot	<i>45.0</i>	28.0
	RT 5-shot	44.0	<i>45.0</i>
awardWonBy	ReWiSe 5-shot	7.5	7.3
	RT 0-shot	0.0	8.0
	RT 5-shot	<i>7.9</i>	7.4
Overall	ReWiSe 5-shot	37.8	41.7
	RT 0-shot	32.2	29.4
	RT 5-shot	32.9	<i>41.3</i>

Per-relation trends are consistent with the intuition from the ablations:

- `countryLandBordersCountry` is handled very well by all three methods once self-consistency is active. It also benefits a lot from CoT.
- For `companyTradesAtStockExchange` and `personHasCityOfDeath`, ReWiSe retains an advantage, suggesting that few-shot reasoning patterns from SyntheticCoT are more helpful than a unified soft prompt.
- `hasArea` and `hasCapacity` remain numerically tricky for all methods; differences are modest.

- For `awardWonBy`, all methods operate in a low Macro-F1 regime; the sub-queries strategy of Section 4.2.3 is better suited to address this relation.

At the aggregate level, Table 4.6 shows that Relation Tuning with zero-shot inference remains below the best ReWiSe configuration on Dataset2025, both with and without CoT. The gap is modest on some relations, such as `countryLandBordersCountry`, and larger on others, notably `companyTradesAtStockExchange` and `personHasCityOfDeath`. This confirms that, in this data-scarce regime, prompt-based reasoning with SyntheticCoT and relation-wise aggregation remains the strongest option among the methods considered.

4.2.5 Experiments on the unified dataset

Objective

The unified dataset combines all editions of the LM-KBC challenge from 2022 to 2025. The objective of this experiment is to test whether the conclusions drawn on Dataset2025 transfer to a larger and more heterogeneous set of relations and subjects. In particular, we compare ReWiSe and Relation Tuning in this broader setting and identify relation types where one approach is clearly preferable.

The main difference with Dataset2025 is how we construct SyntheticCoT. On the unified dataset, SyntheticCoT is generated from HumanCoT examples that do not share the same relation: human-authored reasoning is no longer tailored per relation, and the model is responsible for adapting generic reasoning patterns to new relations. Comparing CoT and non-CoT regimes on the unified dataset and on Dataset2025 therefore tests whether this cross-relation distillation of reasoning schemes generalizes to relations for which we have training examples but no relation-specific gold CoT.

Protocol

The unified dataset is built as described in Section 4.1, by taking the union of the train, validation and test splits across all editions and removing duplicates and cross-split leaks. As for Dataset2025, the ground truth for the test splits remains private, so all evaluations in this experiment are carried out on the unified validation split.

SyntheticCoT for the unified dataset. We extend the HumanCoT and SyntheticCoT construction procedure from Sections 3.2.2 and 4.1.4 to relations from earlier editions, but without writing new HumanCoT per relation. Instead of providing HumanCoT examples

that share the same relation as the target triple, we reuse HumanCoT examples written for Dataset2025 relations as generic reasoning templates. For each training triple in the unified dataset, we sample HumanCoT examples from other relations, use them as few-shot exemplars, and generate SyntheticCoT with the same prompting scheme and filtering strategy as for Dataset2025. Summary statistics for this unified SyntheticCoT dataset are reported in Table 4.7.

Aggregation strategies. For each relation in the unified dataset, we rerun the aggregation-selection procedure of Section 4.2.1 on the unified validation split, using $n_c = 20$ and the same search space over thresholds and numerical operators. The resulting per-relation strategies are then fixed and reused for all methods compared in this experiment.

Compared methods. On the unified validation split, we consider both CoT and non-CoT regimes for ReWiSe and Relation Tuning. Concretely, we compare:

- ReWiSe without CoT: five-shot prompting on the competition train set, $n_c = 20$, relation-wise aggregation.
- ReWiSe with CoT: five-shot prompting on SyntheticCoT, $n_c = 20$, relation-wise aggregation.
- Relation Tuning without CoT: unified prompt tuning trained on the competition train set as in Section 3.4, soft prompt prepended at inference, $n_c = 20$, relation-wise aggregation.
- Relation Tuning with CoT: unified prompt tuning trained on SyntheticCoT as in Section 3.4, soft prompt prepended at inference, $n_c = 20$, relation-wise aggregation.

All methods use the same base model (Qwen3-8B) and the same system prompts and hyperparameters as in Section 4.1, with the only difference being the presence or absence of CoT supervision and the use of trained soft prompts for Relation Tuning.

Metrics and reported quantities

We evaluate performance on the unified validation split using Macro-F1 under the 2025 metric. We report:

- global Macro-F1 for each of the four configurations,

Table 4.7 Relation Statistics for the unified SyntheticCoT dataset. It consists CoTs generated by Llama3.3 70B on train examples of the unified dataset. It uses 2-shot prompting over HumanCoT examples that don't share the same relation. CoTs leading to correct or incomplete outputs are kept, totaling 12,420 examples across 27 relations.

Relation	Total	Correct	Incomplete
BandHasMember	224	124	100
ChemicalCompoundElement	232	223	9
CityLocatedAtRiver	725	575	150
CompanyHasParentOrganisation	626	585	41
CountryBordersCountry	370	269	101
CountryHasOfficialLanguage	315	264	51
CountryHasStates	94	49	45
FootballerPlaysPosition	340	291	49
PersonCauseOfDeath	748	736	12
PersonHasAutobiography	405	377	28
PersonHasEmployer	487	257	230
PersonHasNoblePrize	486	486	0
PersonHasNumberOfChildren	322	322	0
PersonHasProfession	551	64	487
PersonHasSpouse	465	459	6
PersonPlaysInstrument	772	304	468
PersonSpeaksLanguage	846	628	218
RiverBasinsCountry	813	672	141
StateBordersState	178	63	115
StateSharesBorderState	467	241	226
awardWonBy	6	0	6
companyTradesAtStockExchange	474	444	30
countryLandBordersCountry	589	559	30
hasArea	336	336	0
hasCapacity	14	14	0
personHasCityOfDeath	896	896	0
seriesHasNumberOfEpisodes	639	629	10
Overall	12420	9867	2553

- per-relation Macro-F1, in particular to compare CoT vs non-CoT regimes and ReWiSe vs Relation Tuning on the same relation.

Results

We finally turn to the unified dataset that combines all LM-KBC editions from 2022 to 2025 (Section 4.2.5). The goal is to test whether the conclusions drawn on Dataset2025 extend to a larger and more heterogeneous set of relations. In addition, SyntheticCoT is built in a slightly different way here. We have not written HumanCoT example for the new relations, so the process building SyntheticCoT from HumanCoT uses examples from other relations. This only makes a difference on the CoT version of each method. Therefore, a positive impact of CoT would here show that our synthetic data creation is generalizable to a wider set of relations.

A summary table of overall Macro-F1 scores on the unified validation split for the main configurations (the reproduction of the baseline, CoT with a single prediction, ReWiSe with and without CoT, Relation Tuning with and without CoT) is reported as Table 4.8. This table indicates, for each configuration, the global Macro-F1.

Table 4.8 Macro-F1 performance on `dataset_all` validation set comparing baseline, ReWiSe, and Relation Tuning approaches with and without CoT. All experiments use Qwen3-8B. Prompting experiments (ReWiSe, Baseline, one-prediction CoT) use 5-shot prompting while Relation Tuning uses zero-shot prompting. Baseline uses $n_c = 1$ (single generation). ReWiSe uses $n_c = 20$ with self-consistency. Relation Tuning uses prompt tuning with 10 virtual tokens trained for 10 epochs, then applies $n_c = 20$ self-consistency at inference. Results shown are from the tenth epoch.

Method	No CoT	CoT	n_c
Baseline and one-prediction CoT	30.3	32.0	1
ReWiSe	33.5	35.7	20
Relation Tuning	29.0	30.6	20

To better understand the impact of CoT across different relations in the unified dataset, we analyze the performance differences at the relation level. Figures 4.3 and 4.4 show the difference in Macro-F1 scores (CoT minus no CoT) for each relation, ordered by descending difference, for ReWiSe and Relation Tuning respectively. Green bars indicate relations where CoT improves performance, while red bars show relations where no CoT performs better. For ReWiSe, CoT improves performance on 19 out of 27 relations (70%), with the largest gain of 23.9 points on `PersonSpeaksLanguage`. For Relation Tuning, CoT improves performance on 16 out of 27 relations (59%), with the largest gain of 11.5 points

on `ChemicalCompoundElement`. These results demonstrate that while CoT generally helps across most relations, its effectiveness varies significantly depending on the specific relation and method used.

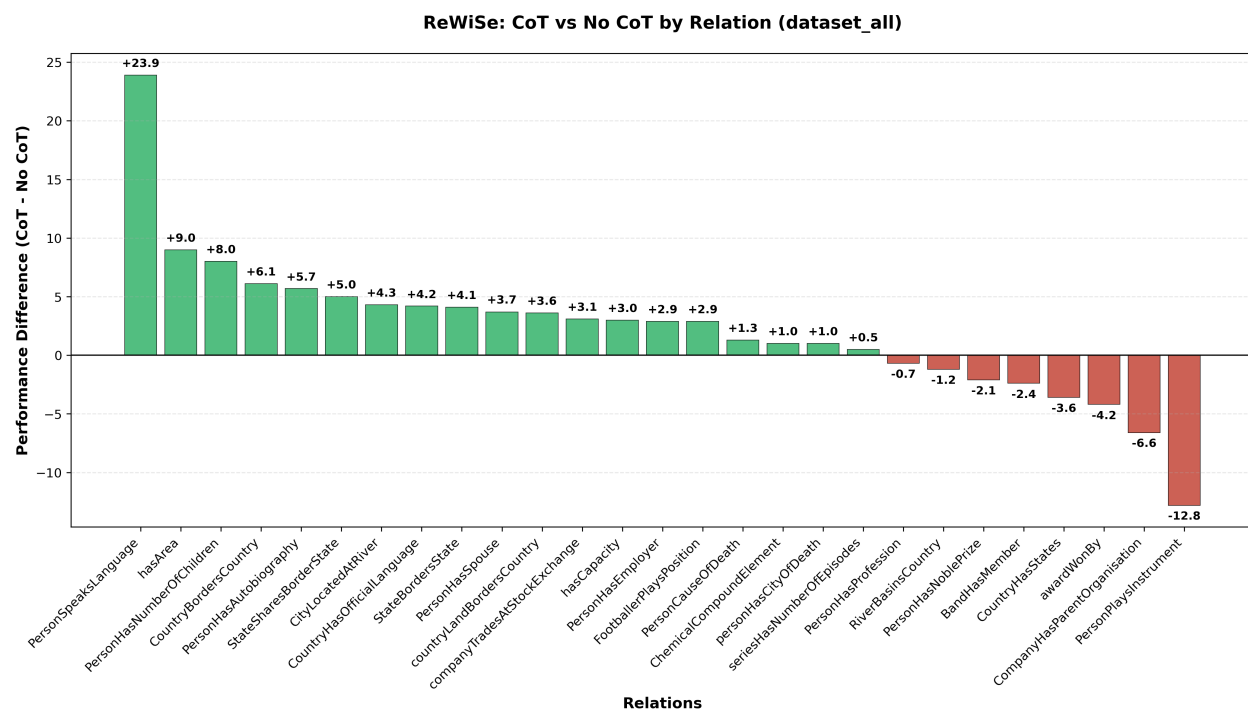


Figure 4.3 Performance difference (CoT - No CoT) in Macro-F1 points difference for each relation using ReWiSe on the unified dataset validation set. Relations are ordered by descending difference.

At the global level, Table 4.8 shows that ReWiSe remains the best method when moving to the unified dataset: it slightly improves Macro-F1 over the baseline in both regimes and maintains a small advantage over Relation Tuning. For all three families of methods (baseline, ReWiSe, Relation Tuning), enabling CoT yields a positive but moderate gain in Macro-F1 at the dataset level. The per-relation analyses in Figures 4.3 and 4.4 indicate that these global gains come from a majority of relations rather than a single outlier, while a minority of relations see small decreases with CoT.

4.3 Discussion

4.3.1 The role of chain-of-thought reasoning

Across all experiments, CoT plays a secondary but consistent role compared to self-consistency. On Dataset2025, moving from single predictions to self-consistency with $n_c = 20$ yields

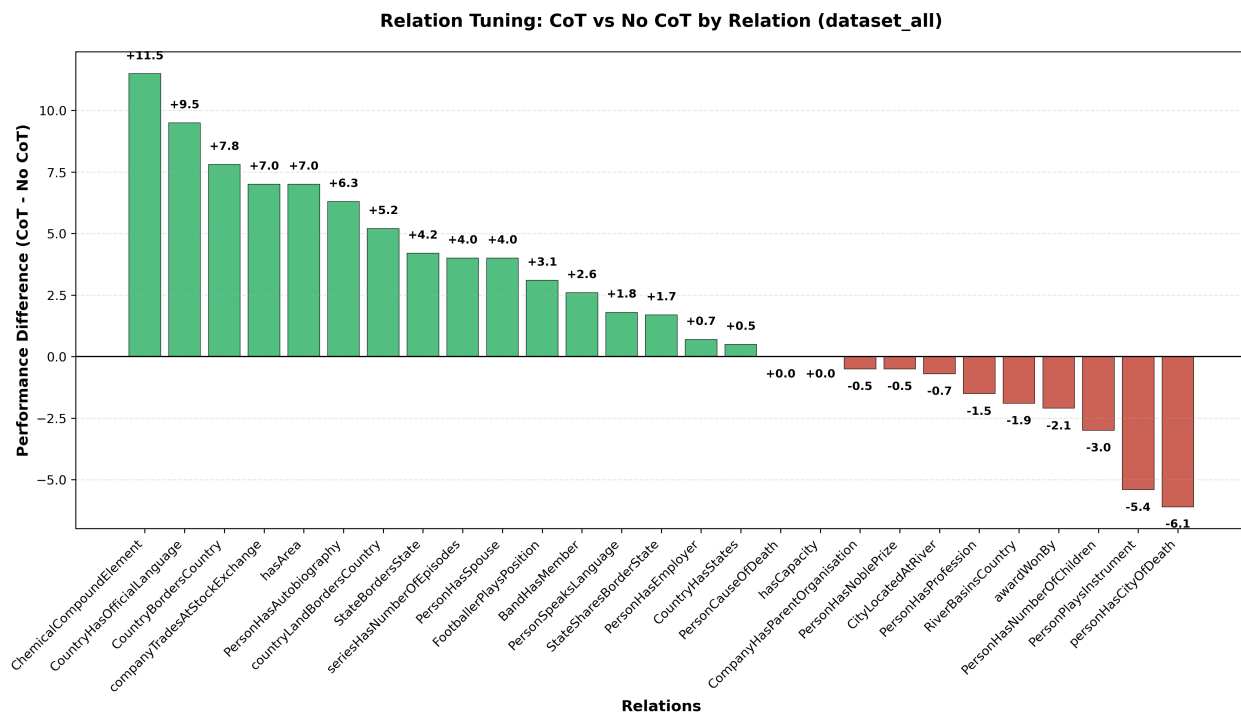


Figure 4.4 Performance difference (CoT - No CoT) in Macro-F1 points difference for each relation using Relation Tuning on the unified dataset validation set. Relations are ordered by descending difference.

the largest performance gains, both with and without CoT. Enabling CoT on top of self-consistency further improves the global Macro-F1, but the magnitude of this gain depends on the relation and CoT hurts the performance on some of the relations.

Relations that carry strong structure or context, such as `countryLandBordersCountry` and `companyTradesAtStockExchange`, benefit the most: CoT helps the model articulate geographic or financial context before committing to an answer set. In these cases, we were able to craft CoTs that guide the model to the production of a better prediction. For numerical relations, CoT has a smaller effect and sometimes only stabilizes behaviour rather than systematically improving scores. The high-cardinality relation `awardWonBy` remains difficult in all regimes, and CoT alone does not resolve its omissions. CoT does not enable the model to produce a long list of outputs

On the unified dataset, the effect of CoT remains positive on average for both ReWiSe and Relation Tuning, even though SyntheticCoT is constructed from HumanCoT examples that do not share the same relation. This suggests that a portion of the reasoning patterns encoded in HumanCoT is relation-agnostic and can be transferred to new relations without dedicated hand-written CoT. At the same time, the per-relation analyses show that this

transfer is uneven: some relations gain substantially from CoT, while others are neutral or slightly negative, indicating that relation-specific reasoning design still matters.

4.3.2 Few-shot prompts and soft prompts

The comparison between ReWiSe and Relation Tuning clarifies the complementary roles of few-shot prompting and prompt tuning. In ReWiSe, few-shot prompts carry three types of information: they enforce the output format, they illustrate the cardinality of typical answers, and they provide miniature reasoning patterns through SyntheticCoT examples. Diversity arises both from the stochastic generation process and from the random selection of few-shot examples, which leads to varied answer sets that can be exploited by relation-wise aggregation.

Relation Tuning replaces explicit few-shot examples with a small set of trainable soft tokens. These tokens encode task information and, depending on the training data, may also encode implicit reasoning patterns or formatting conventions. This makes the inference prompt simpler and avoids sampling variability in example selection. However, once the soft prompt is fixed, the only source of variability is the generation process itself, which tends to reduce the diversity of sampled answers for a given query.

Empirically, Relation Tuning remains competitive but does not surpass ReWiSe in the data-scarce setting of Dataset2025, even though it uses the same base model and the same relation-wise aggregation. This is consistent with the idea that, when training data is limited, explicitly reusing high-quality few-shot examples (especially from SyntheticCoT) can provide more variation and stronger relational cues than encoding everything into a single soft prompt.

4.3.3 Pros and cons of training

Training a Relation Tuning model has both advantages and drawbacks compared to purely prompt-based methods.

On the positive side, prompt tuning provides a compact way to specialize a model to a family of relations without modifying its full parameter set. Once trained, the soft prompt can be reused across many queries without additional design effort and with shorter context which is less expensive at inference time, although the training process is more expensive. On the unified dataset, Relation Tuning with CoT achieves similar averaged results than its non-CoT counterpart but with stronger performance on some of the relations, which is promising, showing that the model can learn to internalize some of the synthetic reasoning patterns.

On the negative side, the gains observed in this work are smaller than those obtained by

tuning the aggregation mechanism and leveraging SyntheticCoT in a purely prompt-based way. The small size of the available training data makes it difficult for the soft prompt to capture all relation-specific nuances, especially for high-cardinality or numerically challenging relations. Moreover, any error pattern present in the training data is baked into the trained soft prompt and will affect all future generations, whereas in ReWiSe such issues can be diluted by varying the few-shot examples and relying on aggregation. This data quality is especially concerning when training on synthetic data since the model will be trained on errors that appear in the CoT if the final answer is correct.

4.3.4 Compute cost

The goal of test-time scaling is to find ways to improve results by using more compute. Our adaptation of self-consistency and sub-query generation both achieve better results on LM-KBC by increasing the amount of generated text. The self-consistency parameter and the number of sub-queries both linearly control this additional inference cost.

Let C denote the cost of a single model generation for one prompt.

With self-consistency, ReWiSe performs n_c generations per query, hence the cost scales as:

$$\text{Cost(ReWiSe)} = n_c \cdot C$$

With sub-queries, we generate n_q sub-queries for each original query and apply self-consistency to each, which yields:

$$\text{Cost(Sub-queries)} = n_q \cdot n_c \cdot C$$

4.3.5 High-cardinality relations and query partitioning

The sub-queries mechanism introduced for `awardWonBy` illustrates a different axis of improvement that is orthogonal to both CoT and training. Instead of trying to recover a long answer list from a single query, the method explicitly decomposes the answer space into several sub-questions whose aggregated predictions are merged at the entity level.

On Dataset2025, this strategy consistently improves Macro-F1 over the corresponding single-query ReWiSe configurations, primarily by increasing recall while keeping precision at a reasonable level. In other words, the model is prompted to pay attention to different regions of the answer space, and relation-wise self-consistency filters out low-consensus entities within each region. This confirms that query design can be used to expose knowledge that is present in the model but not surfaced by a single broad query.

Although this work applies the mechanism only to `awardWonBy`, the formulation is general and could be extended to other high-cardinality relations where answers naturally cluster along time, geography, category or demographic dimensions. In practice, this suggests that explicit query partitioning is a promising complement to both prompt-based and trained approaches for relations with long and heterogeneous answer lists.

4.3.6 Cross-dataset generalization

Finally, in the unified dataset experiments the main observations from Dataset2025 have held which is a favorable indication that the methods we built are relevant for a larger and more heterogeneous collection of relations. Self-consistency and relation-wise aggregation remain beneficial, CoT provides a positive but relation-dependent boost, and ReWiSe keeps a small but stable advantage over Relation Tuning at the global level.

The construction of SyntheticCoT on the unified dataset is deliberately more constrained, since HumanCoT examples are reused across relations rather than being written per relation. The fact that CoT-based configurations still improve Macro-F1 in this setting indicates that even coarse, cross-relation reasoning patterns can be helpful when combined with a robust aggregation mechanism. At the same time, the variability across relations suggests that there is room for more targeted reasoning design or for hybrid approaches that mix a small set of relation-specific HumanCoT examples with more generic templates.

Overall, these results provide preliminary evidence that our conclusions are not specific to Dataset2025, but a more thorough assessment of cross-dataset generalization is still needed. In particular, future work should test the approach on held-out sets of relations that do not appear in the SyntheticCoT construction, on benchmarks with different schemas, and on a larger number of high-cardinality relations where long answer lists are the norm. This would give a clearer picture of how far the proposed methods and synthetic reasoning patterns extend beyond the relations studied here and could be relevant for the automatic construction of an entire KG.

4.3.7 Knowledge limitations

The model may not possess the necessary knowledge to answer certain questions. In such cases, it can either make a guess or acknowledge its lack of knowledge. [4] show that language models are generally well-calibrated on multiple choice and true/false questions " and can assess their own knowledge by asking models to first propose answers, and then to evaluate the probability 'P(True)' that their answers are correct."

However, two problematic scenarios can arise: the model might not know that it doesn't know, leading to assertive but unsupported guesses (hallucinations); or it might think it doesn't know even though it actually does, suggesting that better retrieval strategies or reasoning approaches could help surface this knowledge.

LM-KBC however does not reward systems for acknowledging uncertainty with "I don't know" responses, creating an incentive to always make a prediction. This encourages participants to generate answers even when uncertain. Although our filtering process is designed to mitigate hallucinations in SyntheticCoT—assuming the parametric knowledge between the larger generator model and smaller target model is consistent—the approach still doesn't teach models to recognize the boundaries of their knowledge at inference time and both ReWiSe and Relation Tuning always output a positive answer.

4.3.8 SyntheticCoT quality concerns

Our SyntheticCoT dataset may contain reasoning paths with:

- Wrong information
- Information that the model thinks is false or doesn't know, since it was generated with a different and larger model, which is important for knowledge distillation

It's difficult to determine how the model processes such inconsistencies. Language model hallucinations can snowball: [89] demonstrates that incorrect information in the model's context negatively affects the generation quality.

As our Relation Tuning model is trained on SyntheticCoT, it may learn to sometimes generate wrong facts when generating its CoT. For ReWiSe, this risk is somewhat mitigated as we sample from SyntheticCoT multiple times, meaning that wrong example-related issues won't appear consistently, and the aggregation mechanism may help filter out introduced errors. A critical distinction exists between the two approaches: in Relation Tuning, any erroneous examples in SyntheticCoT potentially influence all generation attempts as they're encoded in the trained parameters, whereas in ReWiSe, such errors would only impact the specific subset of generation attempts where those examples happen to be sampled for few-shot prompting.

CHAPTER 5 CONCLUSION

5.1 Summary of Contributions

This thesis examined how to extract structured knowledge from a LLM’s parametric memory and contributed a practical, schema-aware approach to knowledge base construction. First, we constructed HumanCoT, a small set of hand-authored reasoning patterns designed for systematic reasoning on each relation, and SyntheticCoT, a larger collection of generated rationales paired with answers; together these resources standardize how the model proceeds from a head–relation pair to tails without altering model weights. Second, we introduced ReWiSe, an adaptation of self-consistency that uses relation knowledge. ReWiSe improves on the baseline by implementing both sequential and parallel test-time scaling methods: using additional compute to make better predictions. The sequential method is systematic reasoning paths. The parallel method used is an adaptation of self-consistency’s aggregation procedure which happens at the entity level and follows a strategy that is tuned for each relation on the validation set. Finally, we explored parameter-efficient fine-tuning on SyntheticCoT to assess whether light training can further improve knowledge probing in data-scarce conditions; gains were uneven across relations, clarifying when training is less effective than prompt-based test-time scaling.

5.2 Limitations

The study is confined to closed-book probing. Results therefore characterize the readout of stored knowledge rather than the best attainable accuracy with external evidence. ReWiSe relies on carefully designed relation-wise templates; while compact, these still introduce a manual component and may not transfer unchanged to a new schema. The process of crafting these templates is based on heuristics and we have no way to create an optimal strategy given a relation. Test-time scaling increases inference cost and latency. SyntheticCoT reflects the base model’s capabilities and biases; reasoning traces can be incorrect yet lead to a correct answer. This case may be problematic since the presence of such examples means that prompting or training could happen on incorrect information which could hurt the model’s ability to retrieve facts from its parametric memory. Our exploratory PEFT results are inconclusive across relations and may be sensitive to hyperparameters, initialization, and supervision quality.

5.3 Future Research

High-quality SyntheticCoT is central to the approach as this dataset gives the model instances of the reasoning strategy to follow. A first priority is scale: enlarge the dataset by covering more relations, increasing the number of examples per relation, and introducing reformulations to diversify prompts and rationales. Training and prompting procedures yield should lead to larger gains with a larger coverage and diversity.

A second priority is to have high-quality data. In this thesis, we worked under the assumption that reasoning paths that lead to a correct answer are correct but a further step would control the reasoning paths themselves. This could be done with independent CoT verification or by confronting multiple CoTs attached to the same query. Methods such as recursive self-aggregation [73] can combine and consolidate multiple traces using more compute with more reformulation and combination of reasoning paths. The relevance of this range of ideas for LLM knowledge probing would be interesting to study. As discussed in the thesis, we want to provide the model with examples that comply with the model’s knowledge. A better understanding of knowledge from the inference model on training samples could help clean SyntheticCoT as we might not want to keep correct reasoning traces that contradict the model’s parametric memory.

A third direction is the search for an optimal reasoning strategy. For each relation, we could compare multiple CoT strategies empirically to identify patterns that best align with the target schema. Beyond exhaustive testing, reinforcement learning can be useful to search for prompting strategies [17]. This may require a larger model more data and compute; the motivation is to train the model to produce reasoning traces helping it better retrieve knowledge from its parametric memory.

Finally, future work could restrict the retrieval work to the model’s parametric memory or provide probabilities or confidence intervals on facts. This trust aspect is central if we want to build a knowledge base that would be used in practice.

REFERENCES

- [1] A. Hogan, E. Blomqvist, M. Cochez, C. D’amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann, “Knowledge Graphs,” *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, May 2022. [Online]. Available: <https://dl.acm.org/doi/10.1145/3447772>
- [2] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, “Language Models as Knowledge Bases?” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 2463–2473. [Online]. Available: <https://www.aclweb.org/anthology/D19-1250>
- [3] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, “How Can We Know What Language Models Know?” *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, Dec. 2020. [Online]. Available: <https://direct.mit.edu/tacl/article/96460>
- [4] 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, and J. Kaplan, “Language Models (Mostly) Know What They Know,” Nov. 2022, arXiv:2207.05221 [cs]. [Online]. Available: <http://arxiv.org/abs/2207.05221>
- [5] M. Li, Y. Zhao, W. Zhang, S. Li, W. Xie, S.-K. Ng, T.-S. Chua, and Y. Deng, “Knowledge Boundary of Large Language Models: A Survey,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vienna, Austria: Association for Computational Linguistics, 2025, pp. 5131–5157. [Online]. Available: <https://aclanthology.org/2025.acl-long.256>
- [6] J.-C. Kalo, T.-P. Nguyen, S. Razniewski, and B. Zhang, “Lm-kbc challenge @ iswc 2025,” in *4th Semantic Web Challenge on Language Models for Knowledge Base Construction Challenge*, 2025. [Online]. Available: <https://lm-kbc.github.io/challenge2025/>

- [7] 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, and Z. Qiu, “Qwen3 Technical Report,” May 2025, arXiv:2505.09388 [cs]. [Online]. Available: <http://arxiv.org/abs/2505.09388>
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: <http://aclweb.org/anthology/N19-1423>
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010.
- [10] J. Da, R. L. Bras, X. Lu, Y. Choi, and A. Bosselut, “Analyzing Commonsense Emergence in Few-shot Knowledge Models,” *Automated Knowledge Base Construction, AKBC 2021*, Sep. 2021, arXiv:2101.00297 [cs]. [Online]. Available: <http://arxiv.org/abs/2101.00297>
- [11] B. AlKhamissi, M. Li, A. Celikyilmaz, M. Diab, and M. Ghazvininejad, “A Review on Language Models as Knowledge Bases,” Apr. 2022, arXiv:2204.06031 [cs]. [Online]. Available: <http://arxiv.org/abs/2204.06031>
- [12] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, “Large Language Models Struggle to Learn Long-Tail Knowledge,” 2022, version Number: 2. [Online]. Available: <https://arxiv.org/abs/2211.08411>
- [13] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, “When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories,” in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, 2023, pp. 9802–9822. [Online]. Available: <https://aclanthology.org/2023.acl-long.546>

- [14] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS ’20. Red Hook, NY, USA: Curran Associates Inc., 2020, pp. 1877–1901.
- [15] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, “Training language models to follow instructions with human feedback,” in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS ’22. Red Hook, NY, USA: Curran Associates Inc., Nov. 2022, pp. 27730–27744.
- [16] J. Wei, X. Wang, D. Schuurmans, M. Bosma, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language models,” in *Guide Proceedings*, 2022, archive Location: world. [Online]. Available: <https://dl.acm.org/doi/10.5555/3600270.3602070>
- [17] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Xu, H. Ding, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Chen, J. Yuan, J. Tu, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. You, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Zhou, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma,

- Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang, “DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning,” *Nature*, vol. 645, no. 8081, pp. 633–638, Sep. 2025. [Online]. Available: <https://www.nature.com/articles/s41586-025-09422-z>
- [18] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, “Self-Consistency Improves Chain of Thought Reasoning in Language Models,” *ICLR 23*, Mar. 2023, arXiv:2203.11171 [cs]. [Online]. Available: <http://arxiv.org/abs/2203.11171>
- [19] Z. Li and D. Hoiem, “Learning Without Forgetting,” in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, vol. 9908, pp. 614–629, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-46493-0_37
- [20] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang, “An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-Tuning,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 33, pp. 3776–3786, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/11151751/>
- [21] M. Kazemi, S. Mittal, and D. Ramachandran, “Understanding Finetuning for Factual Knowledge Extraction from Language Models,” 2023, publisher: [object Object] Version Number: 1. [Online]. Available: <https://arxiv.org/abs/2301.11293>
- [22] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig, “Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Miami, Florida, USA: Association for Computational Linguistics, 2024, pp. 7765–7784. [Online]. Available: <https://aclanthology.org/2024.emnlp-main.444>
- [23] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, “GPT understands, too,” *AI Open*, vol. 5, pp. 208–215, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666651023000141>
- [24] B. Lester, R. Al-Rfou, and N. Constant, “The Power of Scale for Parameter-Efficient Prompt Tuning,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 3045–3059. [Online]. Available: <https://aclanthology.org/2021.emnlp-main.243>

- [25] X. L. Li and P. Liang, “Prefix-Tuning: Optimizing Continuous Prompts for Generation,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, pp. 4582–4597. [Online]. Available: <https://aclanthology.org/2021.acl-long.353>
- [26] X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang, “P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland: Association for Computational Linguistics, 2022, pp. 61–68. [Online]. Available: <https://aclanthology.org/2022.acl-short.8>
- [27] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. d. Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-Efficient Transfer Learning for NLP.” Long Beach, California: PMLR 97, 2019, arXiv:1902.00751 [cs]. [Online]. Available: <http://arxiv.org/abs/1902.00751>
- [28] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, “AdapterFusion: Non-Destructive Task Composition for Transfer Learning,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, 2021, pp. 487–503. [Online]. Available: <https://aclanthology.org/2021.eacl-main.39>
- [29] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, and others, “Lora: Low-rank adaptation of large language models.” *ICLR*, vol. 1, no. 2, p. 3, 2022.
- [30] T. Detrmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” *Advances in neural information processing systems*, vol. 36, pp. 10 088–10 115, 2023.
- [31] M. Szep, D. Rueckert, R. v. Eisenhart-Rothe, and F. Hinterwimmer, “Fine-tuning Large Language Models with Limited Data: A Survey and Practical Guide,” Oct. 2025, arXiv:2411.09539 [cs]. [Online]. Available: <http://arxiv.org/abs/2411.09539>
- [32] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp, and J. Lehmann, “Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 8825–8845, Dec. 2022, arXiv:2006.13365 [cs]. [Online]. Available: <http://arxiv.org/abs/2006.13365>

- [33] H. Yang, Z. Lin, and M. Zhang, “Rethinking knowledge graph evaluation under the open-world assumption,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 8374–8385, 2022.
- [34] D. Vrandečić and M. Krötzsch, “Wikidata: a free collaborative knowledgebase,” *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, Sep. 2014. [Online]. Available: <https://dl.acm.org/doi/10.1145/2629489>
- [35] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, “A Survey on Knowledge Graphs: Representation, Acquisition and Applications,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, Feb. 2022, arXiv:2002.00388 [cs]. [Online]. Available: <http://arxiv.org/abs/2002.00388>
- [36] J. Li, A. Sun, J. Han, and C. Li, “A Survey on Deep Learning for Named Entity Recognition,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50–70, Jan. 2022, arXiv:1812.09449 [cs]. [Online]. Available: <http://arxiv.org/abs/1812.09449>
- [37] O. Sevgili, A. Shelmanov, M. Arkhipov, A. Panchenko, and C. Biemann, “Neural Entity Linking: A Survey of Models Based on Deep Learning,” *Semantic Web*, vol. 13, no. 3, pp. 527–570, Apr. 2022, arXiv:2006.00575 [cs]. [Online]. Available: <http://arxiv.org/abs/2006.00575>
- [38] K. Detroja, C. Bhensdadia, and B. S. Bhatt, “A survey on Relation Extraction,” *Intelligent Systems with Applications*, vol. 19, p. 200244, Sep. 2023. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2667305323000698>
- [39] A. Piscopo and E. Simperl, “What we talk about when we talk about wikidata quality: a literature survey,” in *Proceedings of the 15th International Symposium on Open Collaboration*. Skövde Sweden: ACM, Aug. 2019, pp. 1–11. [Online]. Available: <https://dl.acm.org/doi/10.1145/3306446.3340822>
- [40] G. Amaral, O. Rodrigues, and E. Simperl, “ProVe: A pipeline for automated provenance verification of knowledge graphs against textual sources,” *Semantic Web*, pp. 1–34, Sep. 2023. [Online]. Available: <https://journals.sagepub.com/doi/full/10.3233/SW-233467>
- [41] H. Paulheim, “Knowledge graph refinement: A survey of approaches and evaluation methods,” *Semantic Web*, vol. 8, no. 3, pp. 489–508, Dec. 2016. [Online]. Available: <https://journals.sagepub.com/doi/full/10.3233/SW-160218>

- [42] C. Wang, X. Liu, and D. Song, “Language Models are Open Knowledge Graphs,” Oct. 2020, arXiv:2010.11967 [cs]. [Online]. Available: <http://arxiv.org/abs/2010.11967>
- [43] R. Cohen, M. Geva, J. Berant, and A. Globerson, “Crawling The Internal Knowledge-Base of Language Models,” in *Findings of the Association for Computational Linguistics: EACL 2023*. Dubrovnik, Croatia: Association for Computational Linguistics, 2023, pp. 1856–1869. [Online]. Available: <https://aclanthology.org/2023.findings-eacl.139>
- [44] Y. Hu, T.-P. Nguyen, S. Ghosh, and S. Razniewski, “Enabling LLM knowledge analysis via extensive materialization,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 16 189–16 202.
- [45] Y. Hu, T.-P. Nguyen, S. Ghosh, M. Müller, and S. Razniewski, “GPTKB v1.5: A Massive Knowledge Base for Exploring Factual LLM Knowledge,” Jul. 2025, arXiv:2507.05740 [cs]. [Online]. Available: <http://arxiv.org/abs/2507.05740>
- [46] B. Zhang and H. Soh, “Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Miami, Florida, USA: Association for Computational Linguistics, 2024, pp. 9820–9836. [Online]. Available: <https://aclanthology.org/2024.emnlp-main.548>
- [47] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating Embeddings for Modeling Multi-relational Data,” in *Advances in Neural Information Processing Systems*, C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf
- [48] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, “RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space,” *International Conference on Learning Representations (ICLR)*, 2019, publisher: [object Object] Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1902.10197>
- [49] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, “Embedding Entities and Relations for Learning and Inference in Knowledge Bases,” *Proceedings of the International Conference on Learning Representations (ICLR)*, Aug. 2015, arXiv:1412.6575 [cs]. [Online]. Available: <http://arxiv.org/abs/1412.6575>

- [50] T. Trouillon, J. Welbl, S. Riedel, Gaussier, and G. Bouchard, “Complex embeddings for simple link prediction,” in *International conference on machine learning*. PMLR, 2016, pp. 2071–2080.
- [51] I. Balažević, C. Allen, and T. M. Hospedales, “TuckER: Tensor Factorization for Knowledge Graph Completion,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5184–5193, arXiv:1901.09590 [cs]. [Online]. Available: <http://arxiv.org/abs/1901.09590>
- [52] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, “Convolutional 2d knowledge graph embeddings,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018, issue: 1.
- [53] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” in *European semantic web conference*. Springer, 2018, pp. 593–607.
- [54] S. Vashishth, S. Sanyal, V. Nitin, and P. P. Talukdar, “Composition-based Multi-Relational Graph Convolutional Networks,” in *ICLR*, 2020.
- [55] L. Yao, C. Mao, and Y. Luo, “KG-BERT: BERT for Knowledge Graph Completion,” Sep. 2019, arXiv:1909.03193 [cs]. [Online]. Available: <http://arxiv.org/abs/1909.03193>
- [56] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, and J. Tang, “KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation,” 2019, publisher: [object Object] Version Number: 3. [Online]. Available: <https://arxiv.org/abs/1911.06136>
- [57] T. Sun, Y. Shao, X. Qiu, Q. Guo, Y. Hu, X. Huang, and Z. Zhang, “CoLAKE: Contextualized Language and Knowledge Embedding,” in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 3660–3670. [Online]. Available: <https://www.aclweb.org/anthology/2020.coling-main.327>
- [58] C. T. Hoyt, M. Berrendorf, M. Galkin, V. Tresp, and B. M. Gyori, “A Unified Framework for Rank-based Evaluation Metrics for Link Prediction in Knowledge Graphs.” The WebConf: arXiv, Apr. 2022, arXiv:2203.07544 [cs]. [Online]. Available: <http://arxiv.org/abs/2203.07544>

- [59] K. Toutanova and D. Chen, “Observed versus latent features for knowledge base and text inference,” in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Beijing, China: Association for Computational Linguistics, 2015, pp. 57–66. [Online]. Available: <http://aclweb.org/anthology/W15-4007>
- [60] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, “Representing Text for Joint Embedding of Text and Knowledge Bases,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1499–1509. [Online]. Available: <http://aclweb.org/anthology/D15-1174>
- [61] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li, “Realistic Re-evaluation of Knowledge Graph Completion Methods: An Experimental Study,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. Portland OR USA: ACM, Jun. 2020, pp. 1995–2010. [Online]. Available: <https://dl.acm.org/doi/10.1145/3318464.3380599>
- [62] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, “Unifying Large Language Models and Knowledge Graphs: A Roadmap,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–20, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/10387715/>
- [63] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu, “ERNIE: Enhanced Representation through Knowledge Integration,” 2019, publisher: [object Object] Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1904.09223>
- [64] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, “From Local to Global: A Graph RAG Approach to Query-Focused Summarization,” Feb. 2025, arXiv:2404.16130 [cs]. [Online]. Available: <http://arxiv.org/abs/2404.16130>
- [65] A. Roberts, C. Raffel, and N. Shazeer, “How Much Knowledge Can You Pack Into the Parameters of a Language Model?” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, pp. 5418–5426. [Online]. Available: <https://www.aclweb.org/anthology/2020.emnlp-main.437>
- [66] B. Heinzerling and K. Inui, “Language Models as Knowledge Bases: On Entity Representations, Storage Capacity, and Paraphrased Queries,” in *Proceedings of*

- the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, 2021, pp. 1772–1791. [Online]. Available: <https://aclanthology.org/2021.eacl-main.153>
- [67] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, “Locating and Editing Factual Associations in GPT,” Jan. 2023, arXiv:2202.05262 [cs]. [Online]. Available: <http://arxiv.org/abs/2202.05262>
- [68] E. Albert-Rouilhac and A. Zouaq, “Rewise: Relation-wise self-consistency for LLM probing,” in *KBC-LM Workshop and LM-KBC Challenge 2025 (KBC-LM+LM-KBC 2025): Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction*, ser. CEUR Workshop Proceedings, S. Razniewski, J. Kalo, D. Islakoğlu, T. Nguyen, and B. Zhang, Eds., vol. 4041. Nara, Japan: CEUR-WS.org, 2025. [Online]. Available: <https://ceur-ws.org/Vol-4041/paper8.pdf>
- [69] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large Language Models are Zero-Shot Reasoners,” Jan. 2023, arXiv:2205.11916 [cs]. [Online]. Available: <http://arxiv.org/abs/2205.11916>
- [70] N. Ho, L. Schmid, and S.-Y. Yun, “Large language models are reasoning teachers,” in *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, 2023, pp. 14 852–14 882.
- [71] E. Zelikman, Y. Wu, J. Mu, and N. Goodman, “Star: Bootstrapping reasoning with reasoning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 15 476–15 488, 2022.
- [72] X. Chen, R. Aksitov, U. Alon, J. Ren, K. Xiao, P. Yin, S. Prakash, C. Sutton, X. Wang, and D. Zhou, “Universal Self-Consistency for Large Language Model Generation,” Nov. 2023, arXiv:2311.17311 [cs]. [Online]. Available: <http://arxiv.org/abs/2311.17311>
- [73] S. Venkatraman, V. Jain, S. Mittal, V. Shah, J. Obando-Ceron, Y. Bengio, B. R. Bartoldson, B. Kaillkhura, G. Lajoie, G. Berseth, N. Malkin, and M. Jain, “Recursive Self-Aggregation Unlocks Deep Thinking in Large Language Models,” *AKBC 2021*, Sep. 2025, arXiv:2509.26626 [cs]. [Online]. Available: <http://arxiv.org/abs/2509.26626>
- [74] A. Taubenfeld, T. Sheffer, E. Ofek, A. Feder, A. Goldstein, Z. Gekhman, and G. Yona, “Confidence Improves Self-Consistency in LLMs,” in *Findings of the Association for*

- Computational Linguistics: ACL 2025*, 2025, pp. 20 090–20 111, arXiv:2502.06233 [cs]. [Online]. Available: <http://arxiv.org/abs/2502.06233>
- [75] S. Amouyal, T. Wolfson, O. Rubin, O. Yoran, J. Herzig, and J. Berant, “QAMPARI: A Benchmark for Open-domain Questions with Many Answers,” in *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, S. Gehrmann, A. Wang, J. Sedoc, E. Clark, K. Dhole, K. R. Chandu, E. Santus, and H. Sedghamiz, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 97–110. [Online]. Available: <https://aclanthology.org/2023.gem-1.9/>
- [76] S. Singhanian, S. Razniewski, and G. Weikum, “Recall Them All: Retrieval-Augmented Language Models for Long Object List Extraction from Long Documents,” Mar. 2025, arXiv:2405.02732 [cs]. [Online]. Available: <http://arxiv.org/abs/2405.02732>
- [77] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, and E. Chi, “Least-to-Most Prompting Enables Complex Reasoning in Large Language Models.” arXiv, Apr. 2023, arXiv:2205.10625 [cs]. [Online]. Available: <http://arxiv.org/abs/2205.10625>
- [78] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal, “Decomposed Prompting: A Modular Approach for Solving Complex Tasks.” arXiv, Apr. 2023, arXiv:2210.02406 [cs]. [Online]. Available: <http://arxiv.org/abs/2210.02406>
- [79] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis, “Measuring and narrowing the compositionality gap in language models,” in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 5687–5711.
- [80] H. Bast and E. Haussmann, “More Accurate Question Answering on Freebase,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. Melbourne Australia: ACM, Oct. 2015, pp. 1431–1440. [Online]. Available: <https://dl.acm.org/doi/10.1145/2806416.2806472>
- [81] P. Manakul, A. Liusie, and M. Gales, “Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models,” in *Proceedings of the 2023 conference on empirical methods in natural language processing*, 2023, pp. 9004–9017.
- [82] M. Yang, E. Huang, L. Zhang, M. Surdeanu, W. Y. Wang, and L. Pan, “How is llm reasoning distracted by irrelevant context? an analysis using a controlled benchmark,” in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 13 340–13 358.

- [83] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi, “Factscore: Fine-grained atomic evaluation of factual precision in long form text generation,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 12 076–12 100.
- [84] Y. Wang, S. Zhao, Z. Wang, H. Huang, M. Fan, Y. Zhang, Z. Wang, H. Wang, and T. Liu, “Strategic Chain-of-Thought: Guiding Accurate Reasoning in LLMs through Strategy Elicitation,” Sep. 2024, arXiv:2409.03271 [cs]. [Online]. Available: <http://arxiv.org/abs/2409.03271>
- [85] J. He and S. Razniewski, “Combining Self-Retrieval-Augmented Generation with Divide-and-Conquer for Language Model-based Knowledge Base Construction,” in *Joint proceedings of KBC-LM and LM-KBC @ ISWC 2025*, 2025.
- [86] A. Creo, C. Guéret, A. Bernardi, A. Janik, and L. Costabello, “Soft thinking: Enhancing knowledge base completion through trainable prompts in the chain of thought,” in *KBC-LM Workshop and LM-KBC Challenge 2025 (KBC-LM+LM-KBC 2025): Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction*, ser. CEUR Workshop Proceedings, S. Razniewski, J. Kalo, D. Islakoğlu, T. Nguyen, and B. Zhang, Eds., vol. 4041. Nara, Japan: CEUR-WS.org, 2025. [Online]. Available: <https://ceur-ws.org/Vol-4041/paper12.pdf>
- [87] I. M. Sam, “Examining 4-bit quantized Qwen3-8B for object prediction @ LM-KBC 2025,” in *KBC-LM Workshop and LM-KBC Challenge 2025 (KBC-LM+LM-KBC 2025): Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction*, ser. CEUR Workshop Proceedings, S. Razniewski, J. Kalo, D. Islakoğlu, T. Nguyen, and B. Zhang, Eds., vol. 4041. Nara, Japan: CEUR-WS.org, 2025. [Online]. Available: <https://ceur-ws.org/Vol-4041/paper10.pdf>
- [88] A. Clay, E. Jiménez-Ruiz, and P. Madhyastha, “Judge, generator, executioner: Utilizing an LLM for KBC,” in *KBC-LM Workshop and LM-KBC Challenge 2025 (KBC-LM+LM-KBC 2025): Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction*, ser. CEUR Workshop Proceedings, S. Razniewski, J. Kalo, D. Islakoğlu, T. Nguyen, and B. Zhang, Eds., vol. 4041. Nara, Japan: CEUR-WS.org, 2025. [Online]. Available: <https://ceur-ws.org/Vol-4041/paper9.pdf>

- [89] M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith, “How language model hallucinations can snowball,” in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML’24, vol. 235. Vienna, Austria: JMLR.org, 2024, pp. 59 670–59 684.

APPENDIX A INFLUENCE OF THE SELF-CONSISTENCY LEVEL

Figure A.1 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 using ReWiSe. 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.

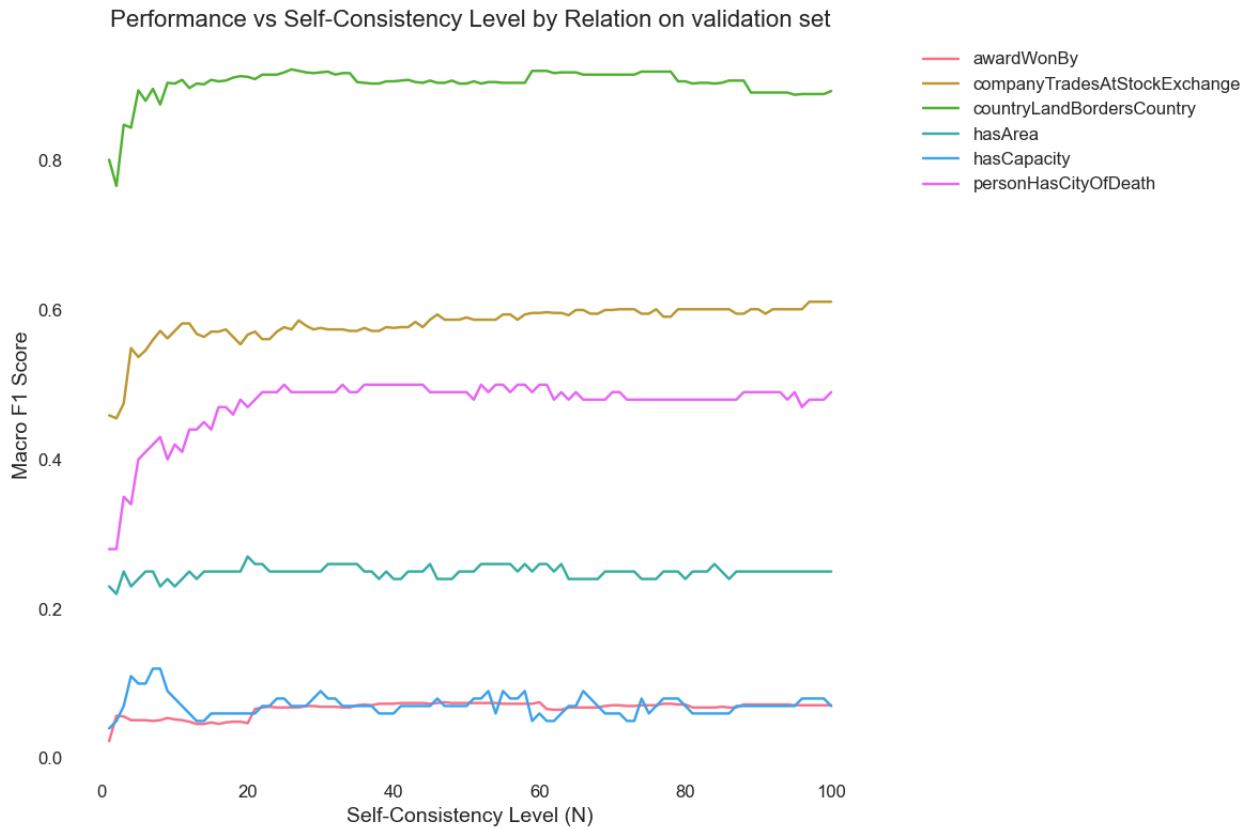


Figure A.1 Macro-F1 score as a function of self-consistency level n_c , on the validation set, split by relation. To compute the prediction at self-consistency level n , we keep the first n predictions over 100.

This plot motivates the choice of $n_c = 20$ across experiments.