T2S-Metrics: Unified Library for Evaluating SPARQL Queries Generated From Natural Language

Yousouf Taghzouti[1,3], Tao Jiang[1], Camille Juigné[2], Benjamin Navet[1,3], Fabien Gandon[2], Franck Michel[2] and Louis-Felix Nothias[1,3]

[1]Univ. Côte d'Azur, CNRS, ICN, France
[2]Univ. Côte d'Azur, Inria, CNRS, I3S, France
[3]Univ. Côte d'Azur, CNRS, Inria, I3S, France

Abstract

The evaluation of Question Answering (QA) systems over Knowledge Graphs has historically suffered from fragmentation, inconsistency, and limited reproducibility. While significant progress has been made in semantic parsing and SPARQL query generation, evaluation methodologies remain diverse, ad hoc, and often incomparable across studies. Existing benchmarks typically focus on a small subset of metrics, such as query exact match or answer-level F1, neglecting syntactic validity, semantic faithfulness, execution correctness, results ranking quality, and computational efficiency. In this paper, we present `t2s-metrics`, an open-source, extensible, and unified evaluation library designed specifically for SPARQL query comparison and execution-based assessment. `t2s-metrics` provides a broad and extensible set of over 20 evaluation metrics, collected from the literature and practical evaluation needs, spanning lexical, syntactic, semantic, structural, execution-based and ranking-based dimensions. These include query-based metrics such as token-level Precision, Recall, and F1; BLEU, ROUGE, METEOR, and CodeBLEU variants; variable-normalized metrics (SP-BLEU, SP-F1); graph- and URI-based exact match metrics; as well as answer set–based metrics such as F1-QALD and Jaccard similarity; ranking metrics including MRR, NDCG, P@k, and Hit@k; and LLM-as-a-Judge metrics. Taking inspiration from the `ir-metrics` library for Information Retrieval, `t2s-metrics` provides a modular abstraction layer that decouples metric specification from implementation, enabling consistent, transparent, and reproducible evaluation of SPARQL-based QA systems. We argue that `t2s-metrics` constitutes a necessary step toward systematic, standardized evaluation in question answering over knowledge graphs and facilitates deeper diagnostic insights into system behavior beyond answer correctness.

Keywords

Question Answering System, Large Language Model, Metric, Text-to-SPARQL

1. Introduction

The translation of natural language questions into formal query languages like SPARQL is a critical component of semantic search and Knowledge Graph Question Answering (KGQA) [1]. As models evolve from template-based systems to end-to-end neural networks and Large Language Models (LLMs), the requirements for evaluation have become increasingly complex [2]. Yet, current evaluation practices are often fragmented. While one study might focus on surface-form metrics like BLEU [3] or ROUGE [4], effectively treating the SPARQL query generation as a machine translation task [5], another might focus strictly on execution accuracy, that is, whether the query retrieves the correct answer [6], while ignoring the syntactic and semantic correctness of the generated query. A third might employ graph isomorphism to check for structural equivalence of two queries [7].

ELMKE: Evaluation of Language Models in Knowledge Engineering 3rd Workshop co-located with ESWC 2026, Dubrovnik, Croatia
yousouf.taghzouti@univ-cotedazur.fr (Y. Taghzouti); tao.jiang@cnrs.fr (T. Jiang); camille.juigne@inria.fr (C. Juigné); benjamin.navet@univ-cotedazur.fr (B. Navet); fabien.gandon@inria.fr (F. Gandon); franck.michel@inria.fr (F. Michel); louis-felix.nothias@cnrs.fr (L. Nothias)
https://youctagh.github.io/ (Y. Taghzouti); https://orcid.org/0000-0002-5293-3916 (T. Jiang); https://cjuigne.github.io/cjuigne/ (C. Juigné); https://github.com/BenjaminNavet (B. Navet); http://fabien.info/ (F. Gandon); https://w3id.org/people/franckmichel (F. Michel); https://lfnothias.github.io/ (L. Nothias)
0000-0003-4509-9537 (Y. Taghzouti); 0000-0002-5293-3916 (T. Jiang); 0000-0003-1157-9030 (C. Juigné); 0009-0005-8523-0941 (B. Navet); 0000-0003-0543-1232 (F. Gandon); 0000-0001-9064-0463 (F. Michel); 0000-0001-6711-6719 (L. Nothias)

This fragmentation makes it difficult to compare results across papers, or to diagnose specific flaws in a model e.g., a model that generates syntactically correct SPARQL but queries non-existing graph patterns. Furthermore, the rise of generative AI has introduced new failure modes, such as "hallucination" [8] where a model makes up plausible-looking but non-existing Uniform Resource Identifiers (URIs).

Traditional metrics like lexical exact match or lexical similarity [9] fail to capture the nuance of two queries that are semantically correct but syntactically distinct due to variations in variable naming or prefix definitions. By contrast, the Information Retrieval (IR) community has long benefited from mature, standardized evaluation libraries [10]. Tools such as `trec_eval` [11] and, more recently, `ir-metrics` [12], provide researchers with consistent access to a wide array of metrics, facilitating reproducibility and fair comparison across systems. Inspired by this paradigm, we argue that KGQA, and SPARQL query generation in particular, requires an equally systematic evaluation framework. To this end, we introduce `t2s-metrics` (where `t2s` stands for text-to-SPARQL), a unified library for evaluating SPARQL queries generated from natural language, designed with four primary goals:

1. **Comprehensiveness:** Support a broad taxonomy of metrics covering lexical similarity, structural equivalence, semantic correctness, execution validity, results ranking quality.
2. **Abstraction:** Decouple metric specification from implementation details, enabling consistent evaluation across datasets and systems.
3. **Diagnosability:** Provide fine-grained signals that allow researchers to understand why a system succeeds or fails with specific metrics, rather than merely whether it does.
4. **Extensibility:** Provide researchers with a framework wherein they can easily add their own metrics and compare them to others.

This paper presents the design principles, metrics taxonomy, and evaluation philosophy underpinning `t2s-metrics`, positioning it as a foundational tool for future KGQA research concerned with evaluation methods, benchmarks, and challenges. More specifically, this paper makes four contributions: (i) a taxonomy of evaluation dimensions for text-to-SPARQL systems, spanning lexical, structural, execution-based, ranking-based, and qualitative assessment perspectives; (ii) a unified evaluation framework that separates metric specification from implementation; (iii) an extensible open-source library, `t2s-metrics`, that operationalizes this framework; and (iv) an empirical illustration showing how multi-metric evaluation provides more informative diagnostics than single-metric assessment alone.

2. Background and Related Works

2.1. Text-to-SPARQL vs IR Evaluation

The evaluation of text-to-code generation sits at the intersection of Natural Language Processing (NLP) and IR [10]. However, evaluating SPARQL-generating QA systems presents challenges that are fundamentally different from traditional IR or text generation tasks:

- **Structural sensitivity:** Minor changes in query structure (e.g., variable placement, OPTIONAL clauses) can drastically affect results.
- **Semantic equivalence:** Two SPARQL queries may be logically equivalent yet lexically dissimilar.
- **Execution dependency:** Query results correctness cannot be assessed independently of the underlying knowledge base, which entails reproducibility issues.
- **Partial correctness:** Queries may retrieve a subset of correct answers, over-generate results, or fail silently (e.g. returning an empty result without raising an error).
- **Ranking and ordering:** Some SPARQL queries produce ranked result sets, for which ranking-aware metrics are necessary.

Various families of metrics have been used over the years:

- **Lexical Overlap Metrics:** Traditionally, systems have been evaluated using metrics borrowed from Machine Translation, such as BLEU [13] and ROUGE [14], also called surface form metrics. While useful for measuring n-gram overlap, these metrics often penalize valid SPARQL queries that use different variable or prefix names, different triple pattern ordering strategies, or semantically equivalent writing alternatives, e.g. using either the FILTER vs. VALUES SPARQL clauses.
- **Execution-Based Metrics:** To address the aforementioned limitations of surface forms, researchers utilize execution accuracy (Precision/Recall of the answer set) [15]. However, this requires a live endpoint with "frozen" dataset, and does not account for queries that are valid but return empty sets intentionally, nor does it penalize inefficient query structures.
- **Semantic and Structural Normalization:** Recent work has highlighted the need for query normalization, renaming variables and expanding prefixes, before comparison. Metrics like SP-BLEU [16] attempt to bridge the gap between surface form and logical structure.

Consequently, no single metric is sufficient to characterize the performance of a system [17].

2.2. Libraries and Tools for Evaluating Text-to-SPARQL Performance

This section overviews libraries and benchmarks relevant to measuring the performance of text-to-SPARQL systems. Where applicable, we note whether the tool provides only a subset of evaluation metrics or lacks direct support for SPARQL query evaluation.

1. **LMs4Text2SPARQL** A codebase for training and evaluating language models on text-to-SPARQL tasks. It includes scripts to compute standard evaluation metrics such as exact match and query execution accuracy.[1]
2. **Text2SPARQL** A repository implementing grammar-pretraining and evaluation pipelines for models that output SPARQL. Includes metrics for SPARQL validity and execution correctness.[2]
3. **LLMs-for-SPARQL** A collection of datasets and evaluation scripts to compare multiple text-to-SPARQL models (e.g., benchmark results across LC-QuAD, Spider4SPARQL).[3]
4. **text2sparql-client** A Python client to interact with text-to-SPARQL-compliant web endpoints. Useful in evaluating endpoint responses, success rates, and basic execution behaviors, but *not* a full evaluation framework for synthetic metrics like F1.[4]
5. **liita-nl2sparql** A Python package that implements a limited set of evaluation metrics for text-to-SPARQL models (e.g., syntactic validity, execution success). It *does not* provide a comprehensive suite of semantic parsing metrics out of the box.[5]
6. **ir_measures** A Python library for calculating classical information retrieval metrics such as precision, recall, and nDCG. It is *not* designed for SPARQL evaluation and thus does not natively handle SPARQL query results or measures like query execution correctness.[6]
7. **pytrec_eval** A Python toolkit for computing evaluation measures from ranked retrieval results (e.g., for TREC-style ad-hoc search). Like `ir_measures`, it provides IR metrics but *does not* support SPARQL or semantic parsing outputs directly.[7]

`t2s-metrics` builds upon these foundations by aggregating these diverse methodologies into a single library. It complements dataset providers like QALD or LC-QuAD by focusing specifically on the computation of evaluation metrics.

[1] https://github.com/AKSW/LMs4Text2SPARQL/blob/main/eval.py
[2] https://github.com/cskyan/Text2SPARQL/blob/master/utils/spider_evaluation.py
[3] https://github.com/SandroGT/LLMs-for-SPARQL/blob/main/experiments/evaluation/metrics.py
[4] https://github.com/AKSW/text2sparql-client/blob/main/text2sparql_client/utils/evaluation_metrics.py
[5] https://github.com/tonazzog/nl2sparql/blob/main/nl2sparql/evaluation/evaluate.py
[6] https://github.com/terrierteam/ir_measures
[7] https://github.com/cvangysel/pytrec_eval/tree/master/py

3. The `t2s-metrics` Library

The `t2s-metrics` Python library provides access to a broad and extensible suite of evaluation metrics tailored to SPARQL generation and KGQA. Table 1 provides a summary of the supported metrics, categorized by their evaluation focus: Lexical, Semantic/Structural, Execution, and Model Performance. A column is dedicated to exemplifying the usage of the metric in literature. Note that Ollama[8] is supported out of the box for local evaluation of the LLM-as-a-Judge metric, with a system evaluator prompt that can be edited if needed[9]. By default the system prompts the LLM to act as a SPARQL expert, scoring a given query from 0.0 to 1.0 based on correctness, efficiency, readability, and adherence to best practices. The LLM returns a structured JSON response containing both a numerical score and a textual reasoning explanation. The method parses this response, handles errors gracefully (defaulting to a score of 0 if parsing fails), and returns a dictionary with the score, reasoning, and raw output.

4. Interfaces and Implementation

`t2s-metrics` is implemented in Python and designed to be easily integrated into training loops or post-hoc evaluation pipelines. The library's **Python API** provides users with a natural syntax to compute metrics, allowing for the seamless calculation of multiple metrics in a single pass. For example, computing SP-BLEU or F1-QALD requires only passing the predicted and gold-standard SPARQL queries, `t2s-metrics` handles the necessary pre-processing such as IRI normalization or variable renaming, automatically.

4.1. Usage

The example in Listing 1 demonstrates evaluating a set of predicted SPARQL queries against ground truth queries, requesting several lexical and execution-based metrics. The predicted and ground truth SPARQL queries are given in a file formatted as shown in Listing 2.

Listing 1: How to use `t2s-metrics` using the Python API

```
from t2smetrics import run_experiments
from t2smetrics.metrics import ( LevenshteinDistance, AnswerSetF1, AnswerSetPrecision,
  AnswerSetRecall, Bleu, CodeBLEU, SPF1, CosineSimilarity, EuclideanDistance, F1Spinach,
  HitAtK, JaccardSimilarity, Meteor, F1QALD, MRR, NDCG, SPF1, RougeN, PrecisionAtK,
  PrecisionQALD, QueryExactMatch, RecallQALD, QueryExecution, LLMJudge, SPBleu, TokenF1,
  TokenPrecision, TokenRecall, URIHallucination )

dataset_name = "example"
graph_path = "./datasets/example/kg/example.ttl"
jsonl_paths = ["./datasets/example/eval/example.jsonl"]
metrics = [
  AnswerSetPrecision(), AnswerSetRecall(), AnswerSetF1(), Bleu(), SPBleu(), CodeBLEU(),
  CosineSimilarity(), EuclideanDistance(), F1QALD(), PrecisionQALD(), RecallQALD(),
  F1Spinach(), HitAtK(k=5), JaccardSimilarity(), LLMJudge(), LevenshteinDistance(),
  MRR(), Meteor(), NDCG(), PrecisionAtK(k=1), QueryExecution(), QueryExactMatch(),
  RougeN(1), RougeN(2), RougeN(3), RougeN(4), TokenF1(), SPF1(), TokenPrecision(),
  TokenRecall(), URIHallucination()
]

run_experiments.run( dataset=dataset_name, jsonl_evals=jsonl_paths, metrics_list=metrics,
  execution_backend_graph_path=graph_path, verbose=True )
```

[8] https://ollama.com/

[9] https://github.com/Wimmics/t2s-metrics/blob/main/t2smetrics/llm/ollama_backend.py

Category	Metric	Description	Usage
Lexical	BLEU@K / BLEU C	Measures k-gram precision against reference. Supports cumulative k-gram aggregation (Geometric mean)	[3]
	SP-BLEU	Adaptation where variables are normalized before standard BLEU calculation	[16]
	CodeBLEU	Evaluates code synthesis quality, incorporating syntactic AST information	[18]
	Cosine Similarity	Calculates the cosine similarity score between the queries by converting them into numerical vectors using a bag-of-words approach	[19]
	Exact Match (Token)	Binary metric (1 or 0) indicating if prediction is identical to reference.	[8]
	F1 (Token)	The harmonic mean of Precision and Recall (AnswerSet)	[2]
	SP-F1	Adaptation where variables are normalized before standard F1 (Token) calculation	[16]
	Jaccard	Intersection over Union of token sets between generated and target queries.	[18]
	METEOR	Matches unigrams based on exact match, stemming, and synonymy.	[20]
	Precision/Recall (Token)	Proportion of correct answers retrieved vs. total retrieved (Precision) or total gold (Recall).	[3]
	ROUGE@K	Assesses overlap of unigrams and bigrams, ect. between model generation and ground truth	[4]
Structural	URI Hallucination	Percentage of queries containing URIs that do not exist in the KB memory.	[8]
Execution	F1 (AnswerSet)	The harmonic mean of Precision and Recall (AnswerSet)	[6]
	F1-QALD	Specialized F1 for Linked Data; handles empty set comparisons specifically	[3]
	F1-Spinach	Specialized F1 for Linked Data; permits additional columns in the predicted results without penalty	[21]
	SP-F1	Adaptation where variables are normalized before standard F1 calculation	[16]
	Query Execution	Measures successful endpoint completion (no errors), acknowledging valid empty sets.	[3]
	Precision/Recall (AnswerSet)	Proportion of correct answers retrieved vs. total retrieved (Precision) or total gold (Recall).	[22]
Ranking	Hit@K	Measures whether at least one relevant item appears in the top K results	[20]
	MRR	Evaluates the position of the correct answer in a ranked list	[23]
	NDCG	Normalized Discounted Cumulative Gain; penalizes correct answers appearing lower in rank.	[24]
	Precision@k	Measures whether the top K results are relevant	[25]
	LLM-as-a-judge	Prompts an LLM to qualitatively judge the query correctness given the question.	[26]

Table 1: Metrics provided by `t2s-metrics` with a paper example using it.

Listing 2: An example of the file containing the predicted and ground truth SPARQL queries used by `t2s-metrics`

```
{"id": "q1", "golden": "SELECT ?x WHERE {?x a <http://example.org/Person>}", "generated":
    "SELECT ?y WHERE {?y a <http://example.org/Person>}", "order_matters": false}
{"id": "q2", "golden": "SELECT ?x WHERE {?x <http://example.org/age>30 }", "generated": "
    SELECT ?x WHERE {?x <http://example.org/age> 40}", "order_matters": true}
```

Normalization Engine. A core component of `t2s-metrics` is the normalization engine. When metrics like SP-BLEU and SP-F1 are requested, the engine automatically:

1. Parses the SPARQL string;
2. Resolves prefixes using a built-in or provided prefix map;
3. Performs variable renaming normalization;
4. Re-serializes the query for comparison.

4.2. Extending `t2s-metrics`

The library can be easily extended with additional metrics. Developers only need to extend the base `Measure` class, or one of the more specialized variants (e.g., `AnswerSetMeasure`), and provide an implementation of the `compute` method, as illustrated in Listing 3.

Listing 3: How to extend `t2s-metrics` with new metrics

```
class MRR(AnswerSetMeasure):
    name = "mrr"

    def compute(self, case, context: EvaluationContext = None):
        gold, pred = self._get_answer_lists(case, context)

        if not self._validate(gold, pred):
            return EvaluationResult(case.id, self.name, 0.0)

        for i, ans in enumerate(pred):
            if ans in gold:
                return EvaluationResult(case.id, self.name, 1.0 / (i + 1))

        return EvaluationResult(case.id, self.name, 0.0)
```

4.3. Source Code Availability

`t2s-metrics` is provided under the GNU Affero General Public License v3.0 or later (AGPL-3.0-or-later) license. The code is published on a public Github repository, and the version used at the time of writing us identified by a DOI to ensure the long-term preservation and citability. The library is published in Pypi. Table 2 summarises the links to the source code, demonstration videos and online prototype of `t2s-metrics`.

License	GNU Affero General Public License v3.0 or later (AGPL-3.0-or-later)
PyPI package	https://pypi.org/project/t2s-metrics/
Repo & DOI	https://github.com/Wimmics/t2s-metrics - 10.5281/zenodo.19255557
Demo video	**Teaser** https://youtu.be/w5nm7Rg5EeM?si **Full** https://youtu.be/ETmzMsnYEqY

Table 2
License and links to the source code, PyPI package, demonstration videos and online prototype of `t2s-metrics`.

5. Practical Use Case and Evaluation

`t2s-metrics` serves two primary user groups:

1. **Model Developers** who need granular feedback. For example, a developer might see high BLEU but low URI EM (Exact Match), indicating their model understands SPARQL syntax but is memorizing/hallucinating incorrect entity identifiers.
2. **Benchmark Creators** who need standard metrics to ensure fair comparison. Using `t2s-metrics`, new datasets can provide baseline scores across 20+ metrics.

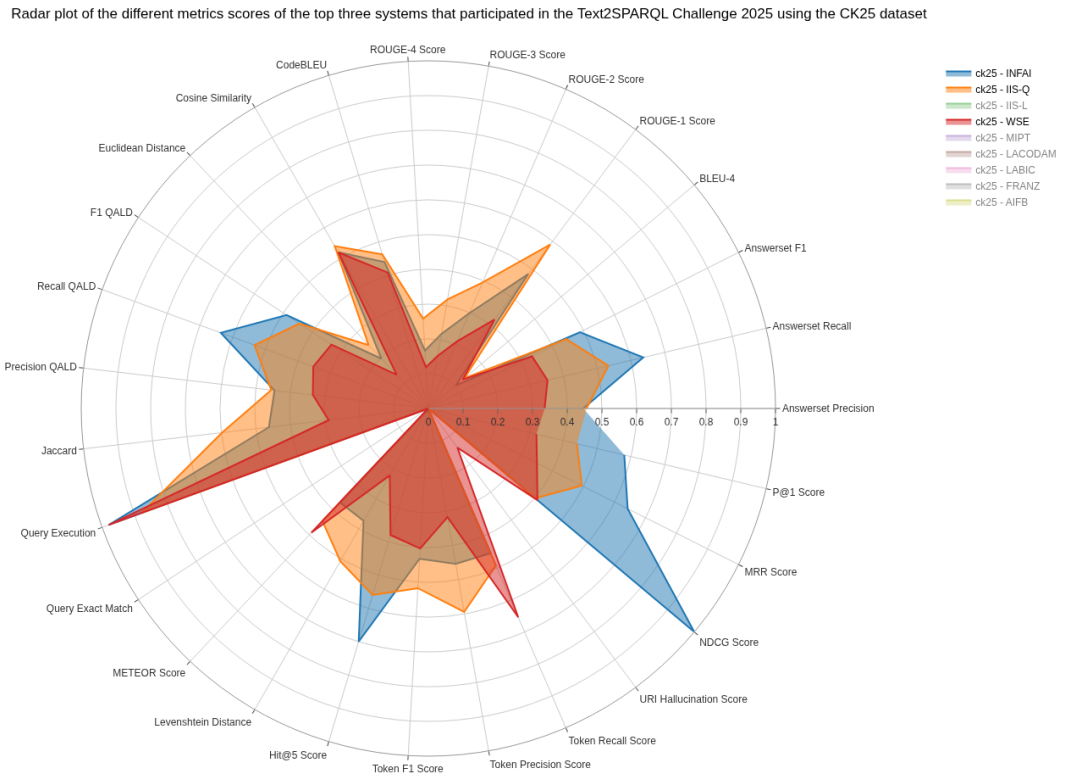

Figure 1: Radar plot of the different metrics scores of the top three systems that participated in the Text2SPARQL Challenge 2025 using the CK25 dataset.

To demonstrate `t2s-metrics`, we used the results produced by the KGQA systems that participated in the Text2SPARQL 2025 challenge,[10] and evaluated them using the metrics we provide on the *ck25* dataset.

Figure 1 presents a radar plot showing the metric scores of the top three systems for the *ck25* dataset, while Figure 2 displays a correlation matrix of the different metrics scores, automatically grouped according to their pairwise correlations. From the correlation matrix, we observe a clear clustering pattern. The bottom-right section shows a concentration of execution-based metrics, whereas the middle section primarily contains string-based metrics (applied to the generated query). Notably, there is a strong negative correlation between the *URI hallucination* metric and the execution-based metrics. This relationship is expected, as an increase in hallucinated URIs often prevents successful query execution, resulting in lower execution scores.

This negative correlation is less pronounced for string-based metrics, with the exception of the *CodeBLEU* metric. This can be explained by the fact that *CodeBLEU* evaluates syntactic correctness by

[10] https://text2sparql.aksw.org/2025/results/

parsing the query, making it more sensitive to errors caused by hallucinated URIs. Since the systems participating in the challenge relied on language models to generate queries, none of their queries matched the gold queries exactly, which explains the zero correlation scores with the *query exact-match* metric.

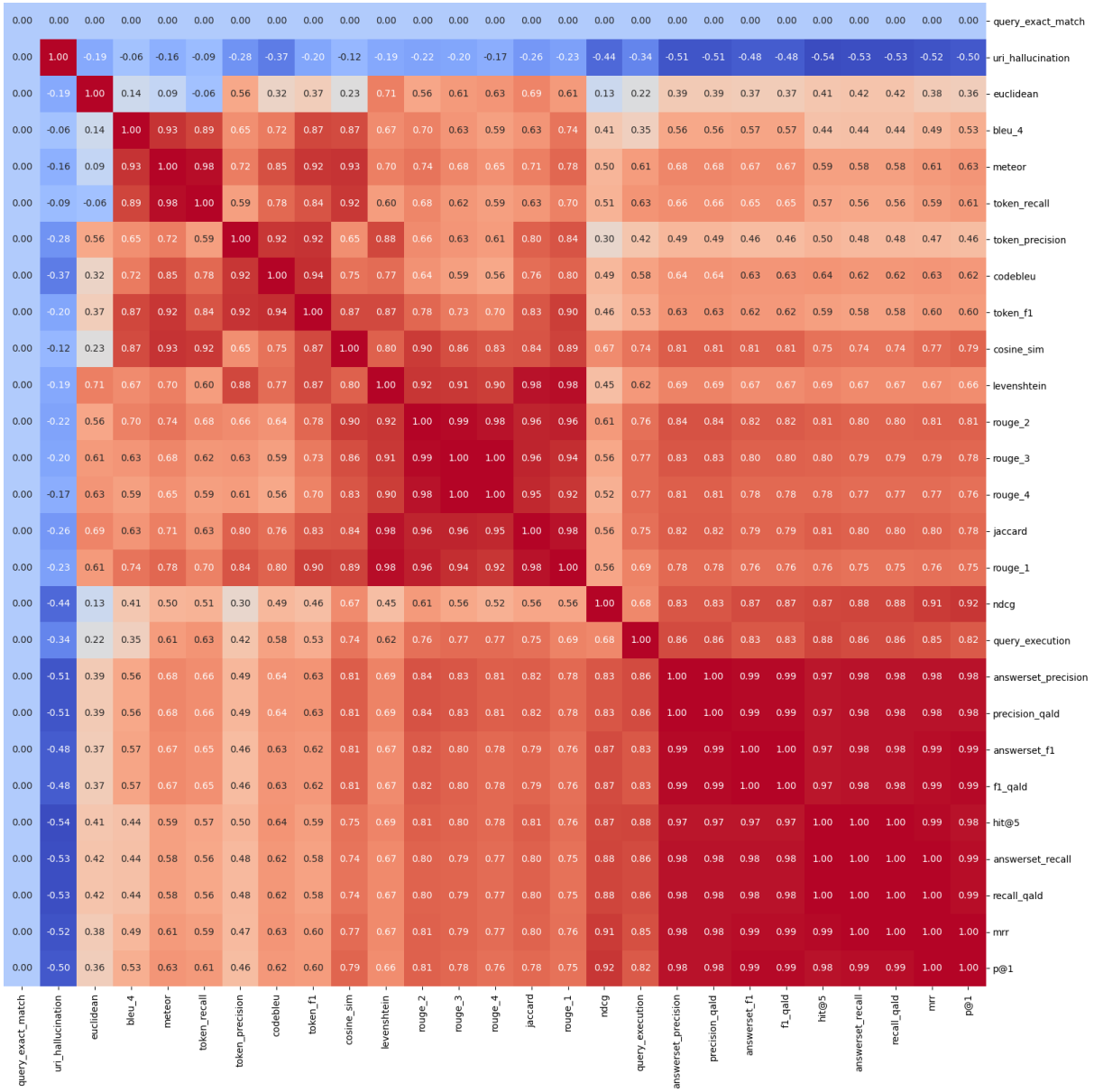

Figure 2: Correlation matrix between the different metrics applied to the systems that participated in the Text2SPARQL Challenge 2025, using the CK25 dataset.

6. Conclusion

We presented `t2s-metrics`, a unified evaluation toolkit for SPARQL generation and KGQA. By aggregating over 20 metrics, including recent adaptations like SP-BLEU and URI Hallucination, `t2s-metrics` addresses the fragmentation in current evaluation methodologies. It allows researchers to move beyond simple text matching and rigorously assess the syntactic validity, semantic accuracy, and execution reliability of their systems. We hope `t2s-metrics` will become a standard utility in the semantic web community, fostering more transparent and reproducible evaluation. We acknowledge that

`t2s-metrics` has limitations in measuring certain operational metrics, including LLM input/output tokens, token pricing, and GPU usage, as these are inherently difficult to compute without access to the internal mechanisms of the underlying methods. As future work, we plan to develop `t2s-metrics` hooks that can be integrated into the implementation to enable reliable capture of these metrics.

Acknowledgments

This work was supported by the French Government's France 2030 investment plan (ANR-22-CPJ2-0048-01), through 3IA Cote d'Azur (ANR-23-IACL-0001) as well as the MetaboLinkAI bilateral project (ANR-24-CE93-0012-01 and SNSF 10002786). Additional support was provided through the UCAJEDI Investments in the Future project (ANR-15-IDEX-01).

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT, Gemini and DeepL for the following: drafting content, grammar and spelling checks. After using these tools/services, the author(s) reviewed and edited the content as needed, taking full responsibility for the publication's content.

References

[1] R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. N. Ngomo, C. Demmler, C. Unger, Benchmarking question answering systems, Semantic Web (2019).
[2] P. A. K. K. Diallo, S. Reyd, A. Zouaq, A Comprehensive Evaluation of Neural SPARQL Query Generation From Natural Language Questions , IEEE Access (2023).
[3] H. M. ZAHERA, M. ALI, M. A. Sherif, D. Moussallem, A. Ngonga, Generating SPARQL from Natural Language Using Chain-of-Thoughts Prompting , International Conference on Semantic Systems (2024).
[4] X. Pan, V. de Boer, J. V. Ossenbruggen, FIRESPARQL: A LLM-based Framework for SPARQL Query Generation over Scholarly Knowledge Graphs , International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (2025).
[5] T. Soru, E. Marx, A. Valdestilhas, D. Esteves, D. Moussallem, G. Publio, Neural Machine Translation for Query Construction and Composition , arXiv.org (2018).
[6] S. Reyd, A. Zouaq, Assessing the Generalization Capabilities of Neural Machine Translation Models for SPARQL Query Generation , International Workshop on the Semantic Web (2023).
[7] Y. Gu, S. E. Kase, M. Vanni, B. M. Sadler, P. Liang, X. Yan, Y. Su, Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases , The Web Conference (2020).
[8] A. Sharma, L. Lara, A. Zouaq, C. Pal, Reducing Hallucinations in Language Model-based SPARQL Query Generation Using Post-Generation Memory Retrieval , arXiv.org (2025).
[9] V. I. Levenshtein, et al., Binary codes capable of correcting deletions, insertions, and reversals 10 (1966) 707–710.
[10] D. Harman, Information retrieval evaluation, Morgan & Claypool Publishers, 2011.
[11] C. Van Gysel, M. de Rijke, Pytrec_eval: An Extremely Fast Python Interface to trec_eval, in: SIGIR, ACM, 2018.
[12] S. MacAvaney, C. Macdonald, I. Ounis, "Streamlining Evaluation with ir-measures", in: Advances in Information Retrieval, Springer International Publishing, Cham, 2022, pp. 305–310.
[13] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a Method for Automatic Evaluation of Machine Translation (2002).
[14] C.-Y. Lin, ROUGE: A Package for Automatic Evaluation of Summaries (2004).
[15] S. Auer, D. Barone, C. Bartz, E. Cortes, M. Y. Jaradeh, O. Karras, M. Koubarakis, D. Mouromtsev, D. Pliukhin, D. Radyush, I. Shilin, M. Stocker, E. Tsalapati, The SciQA Scientific Question Answering Benchmark for Scholarly Knowledge , Scientific Reports (2023).

[16] M. R. A. H. Rony, U. Kumar, R. Teucher, L. Kovriguina, J. Lehmann, SGPT: A Generative Approach for SPARQL Query Generation From Natural Language Questions , IEEE Access (2022).
[17] R. Dividino, G. Gröner, Which of the following SPARQL Queries are Similar? Why? , LD4IE@ISWC (2013).
[18] M. B. Amor, A. Strappazzon, M. Granitzer, E. Egyed-Zsigmond, J. Mitrović, Instruct-to-SPARQL: A text-to-SPARQL dataset for training SPARQL Agents , Conference on Human Information Interaction and Retrieval (2025).
[19] Y. Taghzouti, F. Michel, T. Jiang, L. F. Nothias, F. Gandon, Q²Forge: Minting Competency Questions and SPARQL Queries for Question-Answering Over Knowledge Graphs, in: Proceedings of the 13th Knowledge Capture Conference, 2025.
[20] J. Lehmann, S. Ferré, S. Vahdati, Language Models as Controlled Natural Language Semantic Parsers for Knowledge Graph Question Answering , European Conference on Artificial Intelligence (2023).
[21] S. Liu, S. J. Semnani, H. Triedman, J. Xu, I. D. Zhao, M. S. Lam, SPINACH: SPARQL-Based Information Navigation for Challenging Real-World Questions , Conference on Empirical Methods in Natural Language Processing (2024).
[22] R. Wang, M. Wang, J. Liu, W. Chen, M. Cochez, S. Decker, Leveraging Knowledge Graph Embeddings for Natural Language Question Answering , International Conference on Database Systems for Advanced Applications (2019).
[23] R. Omar, A. Orogat, I. Abdelaziz, O. Mangukiya, P. Kalnis, E. Mansour, Chatty-KG: A Multi-Agent AI System for On-Demand Conversational Question Answering over Knowledge Graphs , arXiv.org (2025).
[24] R. Dorsch, D. Henselmann, A. Harth, Graf von Data: A Knowledge Graph Question Answering Agent for Organisational Usage (2025).
[25] S. Auer, D. Barone, C. Bartz, E. Cortes, M. Y. Jaradeh, O. Karras, M. Koubarakis, D. Mouromtsev, D. Pliukhin, D. Radyush, I. Shilin, M. Stocker, E. Tsalapati, The SciQA Scientific Question Answering Benchmark for Scholarly Knowledge , Scientific Reports (2023).
[26] M. Bekbergenova, L. Pradi, B. Navet, E. Tysinger, F. Michel, M. Feraud, Y. Taghzouti, Y. Z. Chen, O. Kirchhoffer, F. Mehl, et al., MetaboT: An LLM-based Multi-Agent Framework for Interactive Analysis of Mass Spectrometry Metabolomics Knowledge (2025).