Optimization of Methods for Querying Formal Ontologies in Natural Language Using a Neural Network

Anicet Lepetit Ondo[1], Laurence Capus[1] and Mamadou Bousso[2]
[1]*Dep. of Computer and Software Engineering, Laval University, Quebec, G1V 0A6, Quebec, Canada*
[2]*Dep. Computer Science, Iba Der Thiam University, Thies, 967, Senegal*

Keywords: Question-Answering System, SPARQL Language, Transformer, Neural Network, Natural Language Processing.

Abstract: A well-designed ontology must be capable of addressing all the needs it is intended to satisfy. This complex task involves gathering all the potential questions from future users that the ontology should answer in order to respond precisely to these requests. However, variations in the questions asked by users for the same need complicate the interrogation process. Consequently, the use of a question-answering system seems to be a more efficient option for translating user queries into the formal SPARQL language. Current methods face significant challenges, including their reliance on predefined patterns, the quality of models and training data, ontology structure, resource complexity for approaches integrating various techniques, and their sensitivity to linguistic variations for the same user need. To overcome these limitations, we propose an optimal classification approach to classify user queries into corresponding SPARQL query classes. This method uses a neural network based on Transformer encoder-decoder architectures, improving both the understanding and generation of SPARQL queries while better adapting to variations in user queries. We have developed a dataset on estate liquidation and Python programming, built from raw data collected from specialist forums and websites. Two transformer models, GPT-2 and T5, were evaluated, with the basic T5 model obtaining a satisfactory score of 97.22%.

1 INTRODUCTION

A well-designed ontology should be able to answer all user questions. Therefore, it is important to have good coverage. One way to achieve this is by gathering the questions that the ontology should answer and then building the graph accordingly. However, due to the considerable variations in the questions asked by users in natural language expressing same needs, the task becomes complex to extract the meaning of the question and translate it into SPARQL, a formal language to extract the answer from the graph. This involves aligning the concepts present in the user's stated need with those suitable in the triples likely to create a formal query. To meet such a challenge, advanced natural language processing techniques are necessary. Conventional approaches, such as the use of cosine distances or synonym dictionaries, do not always produce the desired results. Moreover, current work faces significant challenges inherent to natural language processing models. Among these challenges are dependency on pre-established patterns, the quality of models and training data, the structure of ontologies, the complexity of resources associated with certain approaches integrating multiple techniques, as well as their sensitivity to linguistic variations for the same need expressed by a user. To optimize this task, we propose in this article a classification method based on the use of Transformers. This method aims to improve the generation of SPARQL queries by promoting better alignment with variations in users' natural language questions. We have experimented with this method on a dataset from two domains: estate settlement and Python programming. The method has also been integrated into a conversational bot to demonstrate its practical feasibility.

The article begins with a review of previous work in this field to situate our contribution. Next, the proposed method is described, including the tools and techniques used. Finally, the results are presented along with their analyses.

Ondo, A., Capus, L. and Bousso, M.
Optimization of Methods for Querying Formal Ontologies in Natural Language Using a Neural Network.
DOI: 10.5220/0012892600003838
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024) - Volume 2: KEOD*, pages 119-126
ISBN: 978-989-758-716-0; ISSN: 2184-3228
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

2 RELATED WORK

A Question-Answering System (or QA System) is a natural language processing (NLP) application designed to automatically respond to questions posed in natural language by users (Nassiri & Akhloufi, 2023). One of the key components of these systems is the dataset used to accomplish the desired task. Several existing datasets utilize a structure where a question is paired with its answer along with context. Others present questions with answers requiring an intermediate step (such as an SQL or SPARQL query) on a knowledge graph or a relational database. These types of datasets are crucial for the development and evaluation of question-answering systems, which aim to optimize querying of ontologies or relational databases in natural language.

Current research in this field is experiencing a significant surge, with sustained efforts to enhance the accuracy and efficiency of responses provided by these systems. For instance, some SQL agents[1] offer a set of more flexible methods to interact with SQL databases, effectively addressing questions based on relational schemas. These agents also allow querying the database as many times as necessary to answer user questions, within the realm of the classical web. However, in the context of the semantic web using ontologies, early studies on SPARQL query generation primarily focus on manual query creation. These studies particularly evaluate the ability of ontology systems to cover and infer within a specific domain, as evidenced by the referenced works (Guo, Pan, & Heflin, 2005; Haase et al., 2010).

With the aim of reducing the need for manual interventions, researchers have focused on SPARQL query generation based on a schema. These schemas function as templates, essentially standardized query forms with reserved slots to fill in (Bagan et al., 2017; Unger et al., 2012). These methods aim to create an intermediate schematic representation of SPARQL queries. Furthering this schema-based query generation logic, other approaches have been developed to associate specific keywords with a domain to create a query model (Zenz et al., 2009). These models are often designed to encapsulate and summarize simple and common SPARQL patterns tailored to a specific dataset. However, there is no guarantee that these models are suitable for other datasets.

New ideas for automatic query generation continue to emerge. A multi-step method translates a user query, expressed in natural language, into a SPARQL query suitable for querying a knowledge graph (Pradel et al., 2013). This process first identifies named entities conveying domain information in the user query. Then, a syntactic dependency analysis is undertaken, followed by the creation of a pivot query. This pivot query explicates the relations inferred from substrings of the user query. Since this pivot query is not directly understandable by the knowledge graph, the last step involves matching it with predefined query templates (which are well-structured queries for querying an ontology) to obtain a list of potential results for the user query. A precise method has been proposed to extract named entities from the user query and then build a dependency tree (Zlatareva & Amin, 2021). SPARQL query generation is based on the relationships present in this tree, associating each found entity or predicate with its syntactic equivalent in the ontology. However, these two previous methods may not be adaptable to other datasets and require substantial technical resources to accomplish this task.

An innovative method proposes integrating the linguistic aspects of questions with graphical data to optimize SPARQL query generation (Rony et al., 2022). By employing a generative model and a stack of Transformer encoders, this method enhances the understanding of natural language questions and facilitates the injection of additional knowledge, such as entities, to generate more precise SPARQL queries. However, errors in predicting relations and entities, as well as variability in formulations, reveal limitations in the model's ability to grasp the subtle nuances of questions and effectively generalize to complex scenarios.

Two machine learning models were utilized (Dileep et al., 2021), namely the random forest classifier and XGBoost, on the LC-QUAD2.0 dataset (Dubey et al., 2019), in comparison with the two Transformer models GPT-2 and T5. It is worth noting that the random forest classifier and XGBoost remain traditional supervised learning models, unlike transformers which are deep learning models revolutionizing classification tasks by utilizing attention mechanisms to capture long-distance dependencies in textual data (Vaswani et al., 2017).

The TSET model was applied to enhance SPARQL query generation by introducing a pre-training step and a Triplet Structure Correction objective (Qi et al., 2024). A semantic transformation method strengthens SPARQL understanding.

[1] https://python.langchain.com/v0.1/docs/use_cases/sql/agents/

Our study highlights the major challenges of natural language processing-based question answering approaches, including reliance on pre-established patterns, model and training data quality, ontology structures, the complexity of resources associated with certain approaches integrating multiple techniques, and sensitivity to linguistic variations for the same user-expressed need. We focus on exploring two transformer models, T5 and GPT2, paying particular attention to effective fine-tuning to accurately predict diverse user inputs expressing the same need.

3 METHODOLOGY

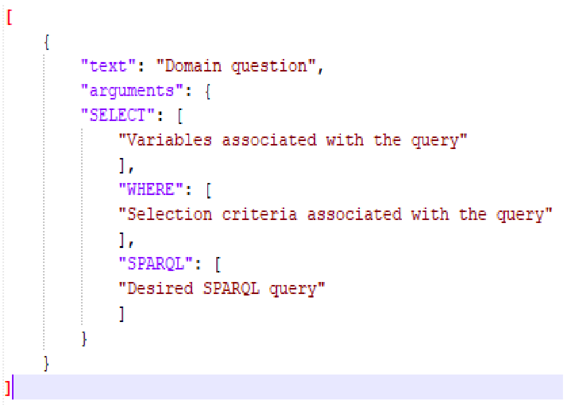
The work presented in this article primarily aims to achieve optimal classification of SPARQL queries based on the varied needs of users for retrieving the same information. User questions and SPARQL queries provided in our dataset are used to train classification models that aim to predict the category or type of SPARQL query associated with a given question. This automates the process of classifying SPARQL queries based on the context of the question.

To accomplish this, two transformer models were experimented with based on the structure of our dataset. Implementing such an approach requires a series of steps from data collection to the obtained results.

3.1 Training Data Collection

Data being essential elements for tasks related to question-answering systems, we chose to test our approach on a dataset covering two domains of expertise: estate settlement and commonly asked questions for learning the basics of the Python programming. To do this, our idea was to gather competency questions (Thiéblin et al, 2021) for these domains to which our ontology should respond, associating them with examples of SPARQL queries related to the entities in our RDF ontology. This implies having a training dataset consisting of pairs of natural language sentences and their equivalents in SPARQL language. These data will be used to teach the model the correspondences between the two languages. We ensured a diversity of examples to cover different categories of queries of unary, binary nature, questions including simple or multiple facts, and temporal aspects.

Figure 1 illustrates the structure found in this dataset, namely a user question associated with various arguments.



```
[
    {
        "text": "Domain question",
        "arguments": {
        "SELECT": [
            "Variables associated with the query"
            ],
            "WHERE": [
            "Selection criteria associated with the query"
            ],
            "SPARQL": [
            "Desired SPARQL query"
            ]
        }
    }
]
```

Figure 1: Structure of the training dataset.

The created dataset includes between three and four paraphrases for each specified need, automatically generated using CHATGPT and also applying crowdsourcing-based (Aydin et al., 2014) from various forums discussing similar topics. This inclusion adds an extra dimension to the variety of questions, thus providing machine learning models with the opportunity to train on diverse formulations of user queries. Consequently, this presents the crucial advantage of avoiding overfitting to a limited set of syntactic variations.

3.2 Models Used

The aim of this work is to encode a user sentence to decode it into a formal SPARQL query capable of querying an ontology. To achieve this, we chose to test two language models, GPT-2 and T5 (Small and Base), pre-trained on our pre-processed dataset. The choice of these Transformer models relies on their main advantage, namely the attention mechanism and multi-head attention (Vaswani et al., 2017), a deep learning principle that selects parts of an input deemed relevant to accomplish the task. This process enables the model to effectively capture complex relationships in data sequences using parallelized computations and by paying particular attention to different parts of the sequence simultaneously. Multi-head attention further enhances this capability by allowing the model to consider multiple perspectives or parameters simultaneously.

3.3 Fine-Tuning

In order to enhance the performance of the models in efficiently generating SPARQL queries related to our OWL ontology, several adjustments have been made to the hyperparameters to stabilize the models, including:

- For the T5 model (Small and Base):
  - ✓ Number of full passes over the training data (8 epochs yielded good performance).
  - ✓ Training batch size set to 2.
  - ✓ Maximum number of checkpoints to keep set to 2.
  - ✓ Frequency at which checkpoints were saved, expressed as the number of training steps, set to 2.
  - ✓ Frequency at which log information was recorded set to 4.
  - ✓ To control the speed at which our model adjusts its weights during training, we set the learning rate to 1e-3.
- For the GPT2 model:
  - ✓ Number of full passes over the training data (our model produced acceptable results at 16 epochs).
  - ✓ To control the speed at which our model adjusts its weights during training, we set the learning rate to 5e-5.

The hyperparameter selection process for each category followed a rigorous methodological approach. This involved a series of successive experiments, during which the hyperparameters were systematically adjusted. After several iterations, only the hyperparameters that ensured the stability and optimal performance of the model were retained.

3.4 Evaluation and Model Results

During the creation of our dataset using paraphrasing mechanisms, we generated multiple paraphrased forms of the same requirement. A portion of these paraphrases was utilized in the training dataset, while the remaining portion was integrated into the test dataset to evaluate our model.

We proceed by describing the evaluation conducted along with the results obtained.

3.4.1 Evaluation Metric

Our evaluation phase involved assessing the quality of the queries generated by the model as well as other accompanying arguments for a SPARQL query using a test dataset. This phase also aimed to ensure that the generated queries are syntactically correct with respect to the entities in the ontology.

Given that our work primarily relies on classification tasks, we adopted the F-measure metric, calculated from precision and recall, to provide an overall assessment of a model's performance. This measure takes into account both the model's ability to correctly identify positive examples (recall) and to avoid classification errors (precision).

Let's recall its general formula:

$$\boldsymbol{F1} = \frac{\boldsymbol{2 * precision * recall}}{\boldsymbol{precision + recall}} \quad (1)$$

With :

$$\boldsymbol{recall} = \frac{\boldsymbol{Truepositive}}{\boldsymbol{Truepositive + falseNegative}} \quad (2)$$

$$\boldsymbol{precision} = \frac{\boldsymbol{Truepositive}}{\boldsymbol{Truepositive + falsePositive}} \quad (3)$$

3.4.2 Results after Training the Models

To train our model, we used a dataset comprising over 1000 questions, each associated with a paraphrased form. The test dataset consisted of 80 questions. Among these, 65 were different reformulations of the questions used during training, and 15 were new questions that had not been used during training.

The queries generated after predicting a user-entered phrase, corresponding to a needs class identified during model training, allow for the extraction of specific information from our ontology used in our approach. These queries are of the 'distinct selection' type and are expressed in the following form:

$$\boldsymbol{\pi_V(\sigma_{condition}(T_1 x\, T_2\, x \ldots x T_n))} \quad (4)$$

- $\boldsymbol{\pi_V}$**:** Represents the projection onto the set of variables V (the columns you want to select).
- $\boldsymbol{\sigma_{condition}}$**:** Represents the selection of triples that satisfy the conditions specified in the WHERE clause, if conditions are present.
- $\boldsymbol{T_1 x\, T_2\, x \ldots x T_n}$ **:** Represents the Cartesian product of $T_1, T_2, \ldots, T_n$, where each T_i is an ontology triplet in the form ⟨s_i, p_i, o_i⟩ for i ∈ {1, 2, …, n}, representing the subject, predicate, and object in the ontology, respectively.

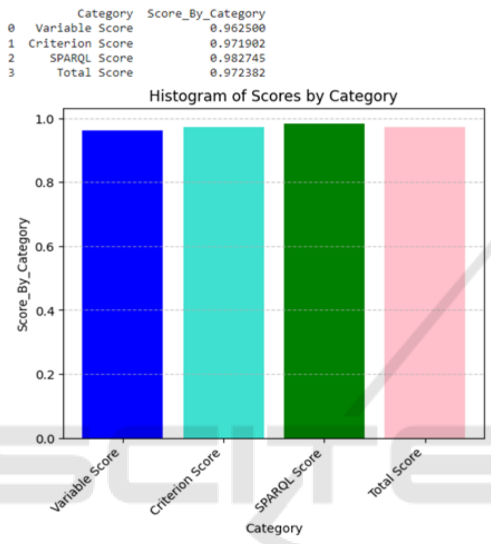
The formal representation that was used and executed within the ontology for formula **(4)** corresponds to the structure of the following SPARQL query:

SELECT DISTINCT V

WHERE $\{T_1 x T_2 x..x T_n\}$ (5)

The results of the tests conducted with the pre-trained models T5-base, T5-small, and GPT2 are illustrated respectively in the figures 2,3 and 4. These results provide an overview of the criteria that were evaluated based on the structure of the dataset. We assessed our approach's ability to generate the complete set of arguments comprising our dataset.

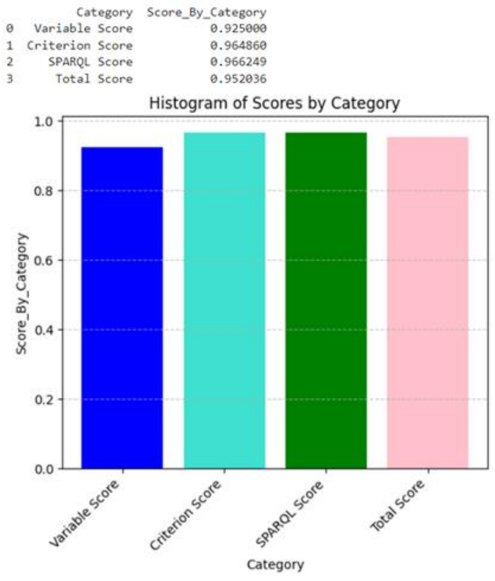
- **For the T5-Base Model**



```
         Category  Score_By_Category
0   Variable Score           0.962500
1  Criterion Score           0.971902
2     SPARQL Score           0.982745
3      Total Score           0.972382
```

Figure 2: Test results with T5-base.

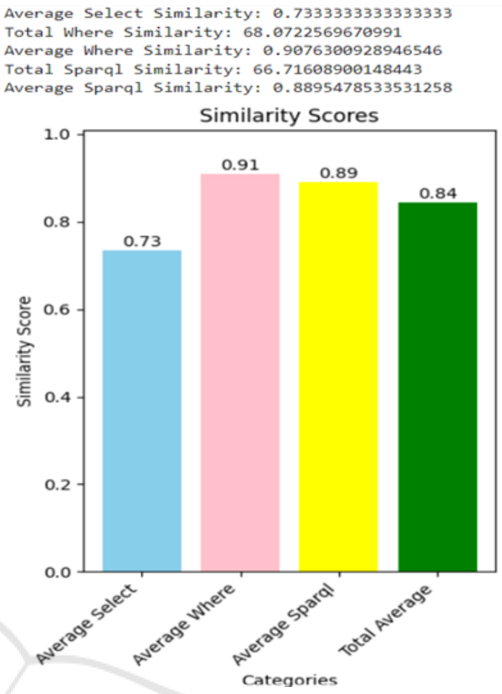
- **For the T5-Small Model**



```
         Category  Score_By_Category
0   Variable Score           0.925000
1  Criterion Score           0.964860
2     SPARQL Score           0.966249
3      Total Score           0.952036
```

Figure 3: Test results with T5-small.

- **For the GPT2 Model**



```
Average Select Similarity: 0.7333333333333333
Total Where Similarity: 68.0722569670991
Average Where Similarity: 0.9076300928946546
Total Sparql Similarity: 66.71608900148443
Average Sparql Similarity: 0.8895478533531258
```

Figure 4: Test results with GPT2.

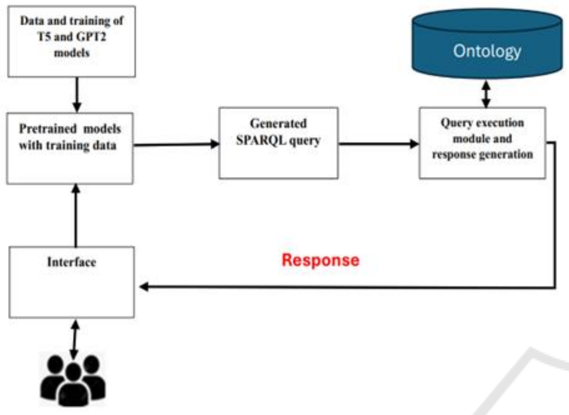
Given that the model's performance in effectively addressing user needs is closely linked to the accuracy of the generated SPARQL queries, we present in Table 1 the F1 score as a combined metric to evaluate the accuracy of the produced responses, thus providing an overall measure of the quality of the SPARQL queries predicted by our approach. This analysis is based on the exact expected values from the test dataset, which consists of 80 questions, and allows for the assessment of the accuracy of the responses generated by our model in relation to the identified correct answers.

Table 1: Score of SPARQL queries likely to interrogate the ontology by tested model.

Models	T5-Base	T5-small	GPT2
Number of questions in the training dataset	1000		
Number of questions in the test dataset	80		
F1 score of the generated SPARQL queries	98,27%	96,62%	88,95%

3.4.3 Implementation of the Approach in a Conversational Bot

Given that the goal of our research is to test transformer encoder-decoder models and experiment with their operation in a conversational system, we propose the resulting architecture presented in Figure 5 below.

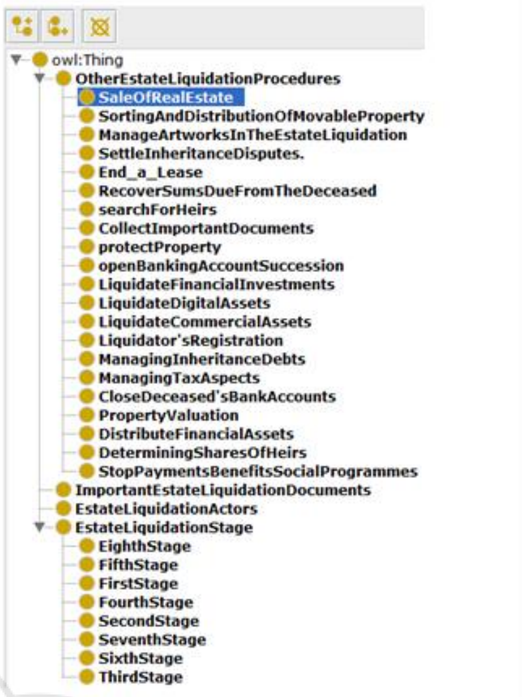


Data and training of T5 and GPT2 models
Ontology
Pretrained models with training data
Generated SPARQL query
Query execution module and response generation
Interface
Response

Figure 5: Technical architecture of the question answering system.

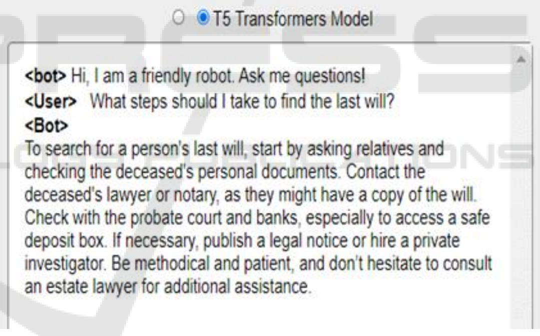
The model providing optimal scores has been deployed in a user-friendly real-time environment, and in a practical manner. We utilized a conversational bot based on Django technology with Python. In order to successfully conduct our experiments, it was essential to test our approach on a formal ontology describing a specified domain in the dataset. The creation of artifacts was done using the Protégé2000 tool from Stanford University for graphical ontology creation, as well as Owlready2 library (Lamy, 2020), allowing the editing of our ontologies using the Python language. To verify the coherence of our ontologies, we used the Hemit reasoner [2] integrated into the Protégé 2000 tool. Figure 6 below presents an excerpt of the ontology artifact used for testing.

Once the training was completed, the model was integrated into our code, allowing to receive the user's sentence as input and translate it into a SPARQL query. This query will then be formatted and enriched with various prefixes to query our ontology, and the obtained response will be returned to the user. The conversational environment was developed using the Django framework, with the shell of a Chatterbot[3] conversational system. Figure 7 shows an example of the result returned by our conversational system.



- owl:Thing
  - OtherEstateLiquidationProcedures
    - SaleOfRealEstate
    - SortingAndDistributionOfMovableProperty
    - ManageArtworksInTheEstateLiquidation
    - SettleInheritanceDisputes.
    - End_a_Lease
    - RecoverSumsDueFromTheDeceased
    - searchForHeirs
    - CollectImportantDocuments
    - protectProperty
    - openBankingAccountSuccession
    - LiquidateFinancialInvestments
    - LiquidateDigitalAssets
    - LiquidateCommercialAssets
    - Liquidator'sRegistration
    - ManagingInheritanceDebts
    - ManagingTaxAspects
    - CloseDeceased'sBankAccounts
    - PropertyValuation
    - DistributeFinancialAssets
    - DeterminingSharesOfHeirs
    - StopPaymentsBenefitsSocialProgrammes
  - ImportantEstateLiquidationDocuments
  - EstateLiquidationActors
  - EstateLiquidationStage
    - EighthStage
    - FifthStage
    - FirstStage
    - FourthStage
    - SecondStage
    - SeventhStage
    - SixthStage
    - ThirdStage

Figure 6: Ontology artifact.



T5 Transformers Model

<bot> Hi, I am a friendly robot. Ask me questions!
<User> What steps should I take to find the last will?
<Bot>
To search for a person's last will, start by asking relatives and checking the deceased's personal documents. Contact the deceased's lawyer or notary, as they might have a copy of the will. Check with the probate court and banks, especially to access a safe deposit box. If necessary, publish a legal notice or hire a private investigator. Be methodical and patient, and don't hesitate to consult an estate lawyer for additional assistance.

Figure 7: Testing the T5 model in a conversational bot.

The result presented in Figure 7 corresponds to the following scenario: A user submitted the sentence, **"What steps should I take to find the last will?"** This sentence was analyzed by our pre-trained T5-base model using specific data. The SPARQL response generated after the analysis was as follows:

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX onto: < http://www.semanticweb.org/test #>
SELECT DISTINCT ?searchLastWill
WHERE

[2] http://www.hermit-reasoner.com/

[3] https://dev.to/documatic/build-a-chatbot-using-python-django-46hb

{ ?lastWill rdf:type onto:CollectImportantDocuments ?lastWill onto:SearchProcedure ?searchLastWill. FILTER (?searchLastWill = onto:lastWill) }

This query was then formatted into an appropriate formal structure and executed at the ontology level, producing the relevant results associated with the concerned subgraphs of the ontology.

4 DISCUSSION AND PERSPECTIVES

It is very challenging, with existing models, to accurately predict a specific class of need, especially when the dataset contains several hundred classes of needs. However, our approach yields better results for user needs classification, thus making a significant contribution to current research. Additionally, we extended our experiments by deploying a chatbot to demonstrate how these tested approaches are utilized in real-time.

In the context of our study, the comparative analysis of the performance of the two considered models, namely T5 and GPT-2, has highlighted significant differences. The results reveal that the T5 model stands out for its remarkable robustness, displaying an impressive score of 97.22%. This outstanding performance remains consistent across various evaluated parameters, validated across an exhaustive range of training tests. In contrast, the GPT-2 model presents less conclusive results, with an overall score of 84%, while showing comparatively lower relative stability. Notably, achieving this level of performance with GPT-2 requires a substantial time investment, as up to 16 training epochs are needed to stabilize its performance, compared to the 8 epochs required by T5 to achieve optimal stability. These empirical observations attest to the superior ability of the T5 model to effectively handle our classification task, by adapting appropriately to the inherent structure of our dataset.

We tested our dataset on a relatively small corpus, aware of the advantages that a richer dataset can provide. With the aim of further enriching this data, we are considering integrating large language models (LLMs), such as GPT-4. Integrating such models will not only enhance the quality of predictions but also generate relevant and fluent responses by leveraging more diverse data during training. Thus, enriching our dataset and adopting more advanced models will help us better address the exponential needs of users and strengthen the robustness of our classification system.

5 CONCLUSIONS

In the scope of our study, we investigated the use of a transformer encoder-decoder model to translate natural language sentences into formal SPARQL queries capable of interacting with an ontology. This approach aims to reduce the complexity of SPARQL language, often used to query RDF ontologies, thereby making these interactions more accessible to users unfamiliar with the syntax of this language.

To accomplish this, we tested two transformer models, T5 and GPT-2, on a dataset composed of pairs of questions and corresponding SPARQL queries, specifically created for this task. The results obtained demonstrate that the T5 model offers superior performance, with a score of 97.22%, and increased stability across various evaluated parameters, requiring only 8 epochs to achieve this stability. In comparison, the GPT-2 model showed lower stability, with a score of 84%, and required up to 16 epochs to achieve comparable results. These findings underscore the robustness and efficiency of the T5 model for generating SPARQL queries from natural language questions, indicating its ability to facilitate interaction with ontologies for non-expert users.

REFERENCES

Aydin, B. I., Yilmaz, Y. S., Li, Y., Li, Q., Gao, J., & Demirbas, M. (2014). Crowdsourcing for Multiple-Choice Question Answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Bagan, G., Bonifati, A., Ciucanu, R., Fletcher, G. H. L., Lemay, A., & Advokaat, N. (2017). GMark: Schema-driven generation of graphs and queries. *IEEE Transactions on Knowledge and Data Engineering*, 29(4), 856-869.

Dileep, A. K., Mishra, A., Mehta, R., Uppal, S., Chakraborty, J., & Bansal, S. K. (2021). Template-based Question-Answering analysis on the LC-QuAD2.0 Dataset. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)* (pp. 443-448). Laguna Hills, CA, USA. doi:10.1109/ICSC50631.2021.00079

Dubey, M., Banerjee, D., Abdelkawi, A., & Lehmann, J. (2019). LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In C. Ghidini et al. (Eds.), *The Semantic Web – ISWC 2019. ISWC 2019. Lecture Notes in Computer Science* (Vol. 11779). Springer, Cham. https://doi.org/10.1007/978-3-030-30796-7_5

Guo, Y., Pan, Z., & Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2), 158-182.

Haase, P., Mathäß, T., & Ziller, M. (2010). An evaluation of approaches to federated query processing over linked data. In *Proceedings of the 6th International Conference on Semantic Systems (I-SEMANTICS)* (pp. 1-9).

Lamy, J.-B. (2020). Ontologies with Python: Programming OWL 2.0 Ontologies with Python and Owlready2. Publisher Apress Berkeley, CA. ISBN 978-1-4842-6551-2. https://doi.org/10.1007/978-1-4842-6552-9

Nassiri, K., & Akhloufi, M. (2023). *Transformer models used for text-based question answering systems*. *Applied Intelligence*, 53, 10602–10635. https://doi.org/10.1007/s10489-022-04052-8

Pradel, Camille & Haemmerlé, Ollivier & Hernandez, Nathalie. (2014). Swip: A Natural Language to SPARQL Interface Implemented with SPARQL. 260-274. 10.1007/978-3-319-08389-6_21.

Qi, J., Su, C., Guo, Z., Wu, L., Shen, Z., Fu, L., Wang, X., & Zhou, C. (2024). Enhancing SPARQL Query Generation for Knowledge Base Question Answering Systems by Learning to Correct Triplets. *Applied Sciences*, 14, 1521. https://doi.org/10.3390/app 14041521

Rony, M. R. A. H., Kumar, U., Teucher, R., Kovriguina, L., & Lehmann, J. (2022). SGPT: A Generative Approach for SPARQL Query Generation From Natural Language Questions. *IEEE Access*, 10, 70712-70723. doi:10.1109/ACCESS.2022.3188714

Thiéblin, É., Haemmerlé, O., & Trojahn, C. (2021). Automatic evaluation of complex alignments: An instance based approach. *Semantic Web*, 12(5), 767–787.

Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., & Cimiano, P. (2012). Template-based question answering over RDF data. In *Proceedings of the 21st International Conference on World Wide Web* (pp. 639-648).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems* (Vol. 30).

Zenz, G., Zhou, X., Minack, E., Siberski, W., & Nejdl, W. (2009). From keywords to semantic queries—Incremental query construction on the semantic web. *Journal of Web Semantics*, 7(3), 166-176.

Zlatareva, N., & Amin, D. (2021). Natural Language to SPARQL Query Builder for Semantic Web Applications. *Journal of Machine Intelligence and Data Science (JMIDS)*, 2. doi:10.11159/ jmids.2021.006