RÉPUBLIQUE DU CAMEROUN
****
Paix - Travail - Patrie
****
UNIVERSITÉ DE YAOUNDÉ 1
****
FACULTÉ DES SCIENCES
****
DÉPARTEMENT D'INFORMATIQUE
****

REPUBLIC OF CAMEROON
****
Peace - Work - Fatherland
****
UNIVERSITY OF YAOUNDÉ 1
****
FACULTY OF SCIENCE
****
DEPARTMENT OF COMPUTER SCIENCE
****

NATURAL LANGUAGE QUESTION ANSWERING OVER KNOWLEDGE GRAPH

Thesis presented in view of obtaining a Master's degree in Computer Science

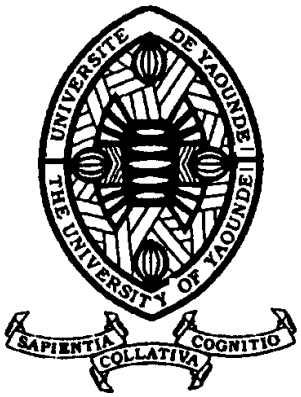
Option : **Génie logiciel**

By: *TSAGUE ESSOMBA VADEL DUMAS*

Matricule: 17T2046

Under the supervision of: Dr JIOMEKONG AZANZI and Pr. GAOUSSOU CAMARA

Academic Year 2022-2023



University of Yaoundé 1

Department of computer science

NATURAL LANGUAGE QUESTION ANSWERING OVER KNOWLEDGE GRAPH

Presented by:

TSAGUE ESSOMBA VADEL DUMAS

Matricule

17T2046

For a Master's degree in Computer Science

Under the supervision of:

Dr. JIOMEKONG AZANZI FIDEL

Pr. GAOUSSOU CAMARA

Academic year 2022-2023

DEDICATION

To my family.

ACKNOWLEDGEMENTS

It is with immense gratitude that I take the pen to express my sincere thanks to all those who have made invaluable contributions to the completion of this Master's thesis.

- First and foremost, I would like to express my deep appreciation to Dr. Azanzi Jiomekong of University of Yaounde I in cameroon and Pr. Gaoussou Camara of University of Alioune Diop de Bambey in Sénégal my thesis advisors, for their guidance, support, and invaluable expertise throughout this demanding academic journey. Your mentorship has been a constant source of inspiration and has greatly enriched my learning experience.
- I would also like to extend warm thanks to the members of the jury for agreeing to read the manuscript and to participate in the defense of this thesis.
- The former heads of Computer Science department Pr. Atsa Roger, Pr. Emvudu Yves; the current head of Computer Science Department Dr. Halidou Aminou for all the facilities and advices they gave to me.
- All the teachers of the Computer Science Department of the University of Yaoundé I.
- I wish to express my gratitude to my classmates and friends who have been an invaluable source of moral support and encouragement throughout this academic journey. Your friendship and solidarity have made this experience even more memorable.
- the persons who contributed from far and near to this thesis.

ABSTRACT

Predicting the type of answer is crucial for providing accurate answers in the automated question-answering system. In the question-answering system, it is essential to know the answer type to reduce the search space of the question. To allow the computer to understand the questions specified in natural language by men and to provide precise answers, it is necessary to go through the prediction of the type beforehand. However given the ambiguity of the questions and the multiplicity of types of answers for the same given question, we are therefore wondering about the establishment of a system for predicting the type of answer in order to achieve an effective reduction of the search space for given questions. Given that, predicting the answer type can be challenging in automated question-answering systems, we have developed a method based on knowledge graph refinement to improve prediction accuracy. Inspired by the TransE algorithm for link prediction on knowledge graph. This method can be applied on any knowledge graph. The popularity of knowledge graphs continues to grow because they allow data to be structured in such a way as to highlight the relationships between entities in a specific domain. It then constitutes the knowledge base of the Answer Type system. The results of our knowledge graph refinement-based approach for answer type prediction have been highly encouraging. Tests conducted showed a significant improvement in prediction accuracy compared to existing methods, thus offering a promising solution to enhance the quality of answers in automated question-answering systems. and we have obtained **96.4%** of accuracy. This method allowed us to win the SMART Task 2022 scientific challenge organized by ISWC.

Keywords — Answer Type Prediction, Knowledge Graph, Knowledge Graph Refinement, Question-Answering, Text Classification, Semantic web Challenge, Dbpedia, Wikidata

RÉSUMÉ

La prédiction du type de réponse est cruciale pour fournir des réponses précises dans les systèmes automatisés de questions-réponses. Elle permet de réduire l'espace de recherche des réponses aux questions, améliorant ainsi la pertinence des résultats. Cependant, compte tenu de l'ambiguïté des questions et de la multiplicité des types de réponses pour une même question, nous nous interrogeons sur la mise en place d'un système de prédiction du type de réponse afin de réduire efficacement l'espace de recherche pour les questions données. Face au défi que représente la prédiction du type de réponse dans les systèmes automatisés de questions-réponses, nous avons développé une méthode basée sur l'affinement d'un graphe de connaissances pour améliorer la précision de cette prédiction. Cette méthode s'inspire de l'algorithme TransE, utilisé pour la prédiction des liens dans les graphes de connaissances, et peut être appliquée à n'importe quel graphe de connaissances. Les graphes de connaissances gagnent en popularité car ils permettent de structurer les données en mettant en évidence les relations entre les entités dans un domaine spécifique, ce qui en fait une base de connaissances idéale pour les systèmes de type de réponse. Les résultats obtenus grâce à notre approche basée sur l'affinement du graphe de connaissances pour la prédiction du type de réponse ont été extrêmement encourageants. Les tests ont montré une amélioration significative de la précision de la prédiction par rapport aux méthodes existantes, ouvrant ainsi la voie à une solution prometteuse pour améliorer la qualité des réponses dans les systèmes automatisés de questions-réponses. En effet, notre méthode a atteint un taux de précision de **96.4%**. Ce travail démontre l'importance de la prédiction du type de réponse et propose une approche efficace pour relever ce défi, ouvrant ainsi de nouvelles perspectives pour l'amélioration des systèmes de questions-réponses automatisés. Cette méthode nous a permis de remporter le challenge scientifique SMART Task 2022 organisé Par ISWC.

Keywords — Prédiction du type de réponse, graphe de connaissances, raffinement du graphe de connaissances, réponse aux questions, classification de texte, défi sémantique du web Dbpedia, Wikidata

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

API Application Programming Interface

AT Answer Type

BERT Bidirectional Encoder Representation from Transformer

CEMAC Central African Economic and Monetary Community

GPT Generative Pre-trained Transformer

GPU Graphics processing Unit

IoT Internet of Thing

ISWC International Semantic Web Conference

KBQA Knowledge Based Question Answering

KG Knowledge Graph

MRR Mean Reciprocal Rank

NDCG Normalized Discounted Cumulative Gain

RDF Resource Description Framework

RNN Recurrent Neural Networks

SMART SeMantic AnsweR Type

SPARQL SPARQL Protocol And RDF Query Language

SVM Support vector Machine

TF-IDF Term frequency - Inverse Document Frequency

URI Uniform resource Identifier

XML Extensible Markup language

GLOSSARY

1. **data Oversampling**: Oversampling involves increasing the number of samples from the minority class (i.e., the class with fewer examples) to reduce the imbalance between classes.
2. **data Augmentation**: Data augmentation is a technique commonly used in the field of machine learning and computer vision. It involves generating new training data from existing data by applying various transformations and modifications to the original dataset.
3. **TransE**: TransE is a machine learning algorithm used in the field of knowledge graph embeddings, specifically in the context of knowledge representation learning. It's designed to learn continuous vector representations of entities and relations in a knowledge graph.
4. **Attention mechanism**: in the context of machine learning and deep learning, is a component or technique that enables a neural network to focus on specific parts of the input data when making predictions.
5. **TF-IDFVectorizer**: Term Frequency-Inverse Document Frequency Vectorizer, is a popular tool in natural language processing (NLP) and information retrieval for converting a collection of text documents into numerical vectors.
6. **Sklearn**: is a popular Python library dedicated to machine learning and data exploration. Scikit-learn provides a set of tools and algorithms for performing a variety of machine learning tasks, including classification, regression, clustering, feature selection, dimensionality reduction, data preprocessing, and more.
7. **SpaCy**: is an open-source natural language processing (NLP) library in Python. It is designed to perform various natural language processing tasks, such as text analysis, named entity recognition, part-of-speech tagging, lemmatization, sentence detection, and more.
8. **Empirical research** is a research method that relies on observation and the collection of real-world data to address research questions or test hypotheses

CHAPTER 1

INTRODUCTION

In our era of ever-expanding information, rapid and accurate access to knowledge has become a necessity. Question-answering systems allow users to interact with vast amounts of available data. However,to provide relevant answers, these systems must not only understand the content and meaning of the posed question but also accurately predict the type of response.

Knowledge Base Question Answering (KBQA) is a popular task in the field of Natural Language Processing and Information Retrieval, in which the goal is to answer a natural language question using the facts in a Knowledge Base. KBQA can involve several subtasks such as entity linking, relation linking, and answer type prediction [1]. In this thesis, we focus on Answer Type subtasks. This prediction of the response type represents a major challenge in the field of natural language processing. By observing this reality, We then ask ourselves the question of **How can we optimize the use of knowledge graphs to predict answer type, considering their inability to directly answer questions, to achieve an effective reduction of the search space for given questions?**

Most approaches to solving this problem are based on classification models. Among these, *Augmentation based answer type classification of the smart dataset* Whose author are *Aleksandr Perevalov and Andreas Both* [2]. They address this problem by proposing a data augmentation technique, and then they create seven classification models to solve it, *Semantic Answer Type Prediction using BERT*IAI at the ISWC SMART Task 2020* Whose author are *Vinay Setty and Krisztian Balog* [3]. They are using Transformer like BERT to solve this problem. However, these approaches have limitations, the most common of which are: Difficulty in modeling the semantic complexity of questions, the high cost of creating datasets, the high implementation cost of these solutions, lack of flexibility to adapt to new domains. In order to overcome some of these limitations, it is therefore important and urgent to firstly explore new approaches that are more efficient And that has a high adaptability to domain change. and secondly augment the repository of solutions to the AT task by proposing new methods [4].

In this thesis, we propose an innovative approach based on knowledge graph refinement to enhance answer type prediction. This approach utilizes the data provided by SMART 2022 Semantic Web Challenge organizers in order to construct and enhance the knowledge graph.

Our work will revolve around three main axes. In Chapter Two, we provide a state of the art on Answer Type Prediction, exploring various resolution approaches and their limitations. In Chapter Three, we will present our approach. Chapter Four will be dedicated to experimentation and the results obtained, where we will describe the evaluation metrics used and the experimental protocol implemented. By following this structure, we hope to make a significant contribution to answer type prediction by leveraging knowledge graphs and proposing effective strategies for reducing the search space.

CHAPTER 2

ANSWER TYPE

2.1 Introduction

Prediction of the answer can be done using several tasks, among which Answer type is one. Answer Type is a crucial subtask in Question Answering in which the main goal is to predict the type of the answer for a given Question ask in a natural language in order to significantly reduce the search space for an answer. **answer type prediction** can be described as follows: Given a question in natural language, the task is to predict type of the answer using a set of candidates from a target ontology [1]. for Example Given these Questions below:

- *where is the university of Yaounde I located ?* The Answer types of this question are: **Location, Place, Populated place, Settlement**.,
- *Who is the president of Cameroon?* the answer types are **Person, Agent**
- *Give me the country where the CEMAC summit 2023 took place ?* in this question, the answer types are **Country, State, Populated place, Place, Location**
- **What year was Ousmane Sonko born?** here the type of the answer for this question is a literal and it is **Date**
- **Can meta-heuristic algorithms be used to solve optimization problems?** The type of the answer to this question is **Boolean**.

The answer type prediction task can be divided into two main subs tasks which are: ***Category Prediction*** and ***Types Prediction***. By knowing the category, we can therefore predict the type of the answer. Categories are: **Resource, Literal, Boolean**: If the category is **resource**, answer types are *ontology classes*. If the category is **literal**, answer types are either *number*, *date*, or *string*. If the category is "Boolean", answer type is always "Boolean".

The ability to know the AT enables the system to significantly narrow down the search space for an answer to a given question [5]. for example considering the sparql query below: cm

```
# without AT prediction
SELECT (COUNT(DISTINCT ?obj)
as ?countObject)
WHERE {
dbr:Charlotte_Dipanda ?p ?obj .
}
# ?countObject = 103
```

```
# with AT prediction
SELECT DISTINCT ?obj as ?count
WHERE {
dbr:Charlotte_Dipanda ?p ?obj .
?obj rdf:type ?type .
FILTER(?type = dbo:Person)
}
# ?count = 6
```

In these queries we want to get all object which interact with the subject **Charlotte Dipanda**. We can see that without knowing the type of the object with which Charlotte Dipanda interacts, the number of responses can explode, whereas knowing the type of object with which she interacts can significantly reduce the search space. For this Question the reduction achieves **94.17%**.

We said that when the question has resource as category, the types are set of classes of an ontology. This ontology can be represented as knowledge graphs, which facilitate the visualization, exploration, and manipulation of these formal models of knowledge representation. Therefore, incorporating knowledge graphs into answer type analysis can offer valuable insights and a deeper understanding of the underlying data.

2.2 knowledge Graphs

A knowledge graph is a structured representation of knowledge. It is used to capture and model information in a specific domain (see fig. 2.1), enabling navigation and exploitation of relationships between entities to obtain in-depth knowledge. Knowledge graphs are commonly used in fields such as artificial intelligence, natural language processing, and information retrieval.

2.2.1 Definition

1. According to **the web semantic journal**, Knowledge graphs are large networks of entities, their semantic types, properties, and relationships between entities [6].
2. According to **the Web Semantic Company** Knowledge graphs could be envisaged as a network of all kind things which are relevant to a specific domain or to an organization. They are not limited to abstract concepts and relations but can also contain instances of things like documents and datasets [6].
3. According to **Farber et al.**, We define a Knowledge Graph as an RDF graph. An RDF graph consists of a set of RDF triples where each RDF triple (s, p, o) is an ordered set of the following RDF terms: a subject $s \in U \cup B$, a predicate p $\in$ U, and an object$U \cup B \cup L$. An RDF term is either a URI $u \in U$, a blank node $b \in B$, or a literal $l \in L$ [6].

In the literature, the third definition is the only formal one, it contradicts with more general definitions as it explicitly requires the RDF data model. In the article [5], the author mentions a difference between the knowledge graph and the ontology by saying: A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge. Another difference is that an ontology is a conceptual model of a knowledge domain. It defines the concepts, relationships, and axioms that are relevant to that domain. or A knowledge graph is a representation of an ontology, it uses nodes

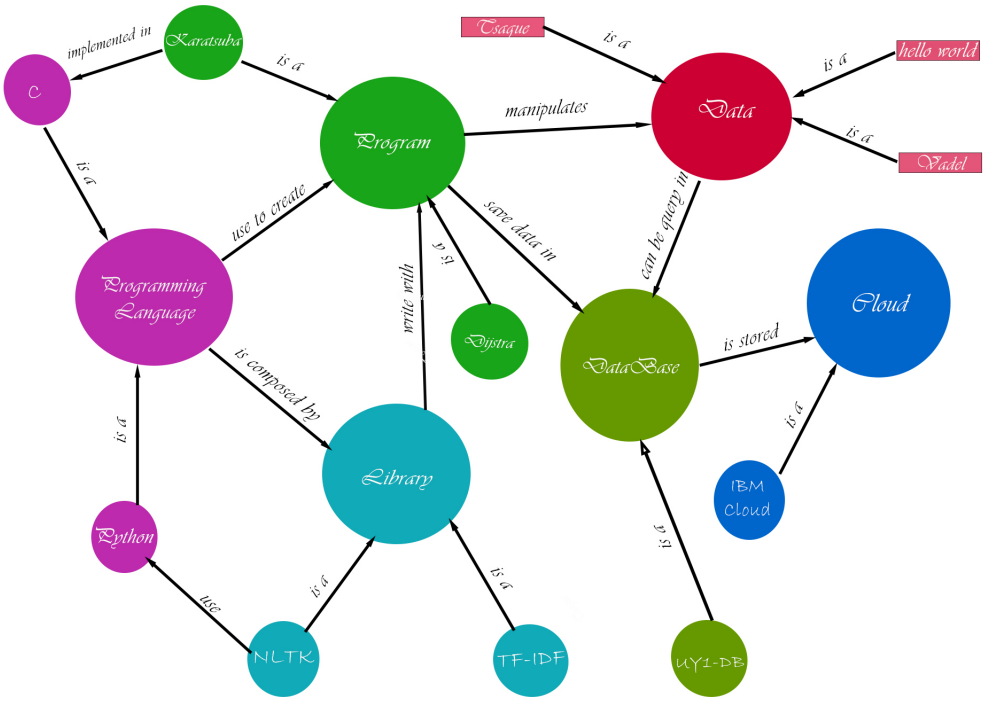


Figure 2.1: Knowledge Graph

to represent concepts and edges to represent relationships. Mere ontologies without instances (such as DOLCE) would not be considered as Knowledge Graph [7]. Now that we have a better understanding of what a knowledge graph is, let's look at some of the most common uses of this technology.

2.2.2 Uses of Knowledge Graph

The knowledge graphs have many uses in a variety of fields, here are some examples of the most common uses

- knowledge graph representation learning [8]: The knowledge graphs can be used to manage knowledge within an organization by creating common ontologies for employees and linking them to relevant information.
- Internet Of Things: a multi-layer knowledge graph is constructed to hierarchically fuse the IoT data [9].
- Information Retrieval: Knowledge graphs can be used to facilitate information retrieval by providing more accurate and relevant results. A study showed that users preferred knowledge graph-based search results over traditional search engine results.
- Semantic Search: Knowledge graphs can be used to improve semantic search, allowing users to ask questions in natural language and receive answers based on structured knowledge. A study showed that knowledge graphs improved the accuracy of semantic search compared to keyword-based approaches.
- Data Analysis: Knowledge graphs can be used for data analysis, allowing researchers to discover hidden patterns and relationships in data. A study showed that knowledge graphs enabled better analysis of research data in life sciences.

- User Experience Improvement: Knowledge graphs can be used to improve the user experience by providing personalized recommendations and helping users navigate content. A study showed that knowledge graphs improved the user experience of digital libraries.

there are some commonly used knowledge graphs:

* ***Wikidata*** Knowledge Graph
* ***DbPedia*** Knowledge Graph
* ***ORKG*** Knowledge Graph
* ***Tsotsa*** Knowledge Graph

2.2.3 Why Knowledge Graph

It is important to note that for many knowledge graphs, one or more refinement steps are applied when creating and/or before publishing the graph. For example, logical reasoning is applied on some knowledge graphs for validating the consistency of statements in the graph and removing the inconsistent statements. Such post processing operations (i.e., operations applied after the initial construction of the graph) would be considered as refinement methods [10].

2.3 Tools Used to Solve AT

In this part we are going to present some commonly used tools for solving the Answer Type Problem. Since 2014, we have seen a significant advancement in the field of natural language processing with the arrival of RNN/LSTM and Transformers. In this section, we will see how RNNs and Transformers revolutionize the resolution of Answer Type problems.

- *RNN*: Recurrent Neural Networks (RNNs) are neural network architectures that have a natural ability to understand sequential data, such as sentences or documents. They can consider sequential dependencies by propagating information from one step to another in the sequence. RNNs have played a significant role in NLP tasks in general, and Answer Type in particular, due to their ability to understand contextual informations, word order, and their dependencies. While RNNs have been widely used for NLP tasks, they have certain limitations. These include **difficulties in parallelization** due to their sequential nature, challenges in capturing **long-term dependencies** when key information is distant within the sequence, and **sensitivity to word order**, resulting in different responses with slight modifications in word order. In 2017, Transformers were introduced to address the most of these limitations.
- *Transformers* are new deep learning architecture which doesn't use sequential processing and hence, have a lot of potential for parallel processing. Most of these models have the **Encoder/Decoder** architecture (see fig. 2.2). They are using **Attention mechanism** [11]. Transformers can process the entire sequence simultaneously, considering long-term contextual dependencies without being limited by sequential order. This enables them to capture complex relationships between words and achieve a finer contextual understanding by enhancing the accuracy of the provided responses. The most commonly models used are :
  - BERT:(Bidirectional Encoder Representations from Transformers): It is a pre-trained model create by Google which excels in Word Masking and the relationship between pairs of sentences.

- T5: (Text-to-Text Transfer Transformer): It is a Transformer model that aims at solving various natural language processing tasks by reformulating them as text-to-text translation problems.
- GPT:(Generative Pre-trained Transformer): This is another pre-trained Transformer model create by OpenAI, it is more used for text generation.

Transformer models, especially large ones like GPT-3 and BERT, require significant computational power for training. They often demand expensive hardware resources, such as high-end GPUs or the use of cloud computing services, which can lead to high costs.

Numerous works utilizing different methodologies to address the problem of Answer Types, based on these advancements, have populated the literature, among which are:

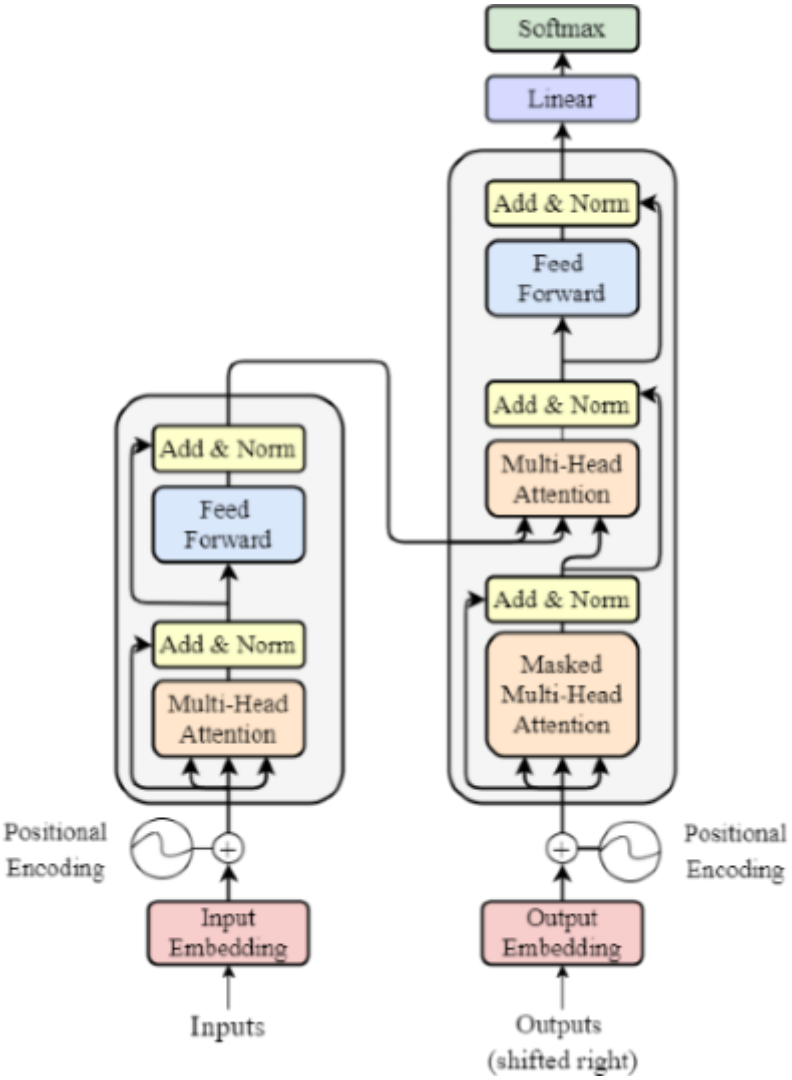


Figure 2.2: Transformers Architecture

2.4 Methodologies Used to Solve AT

Here we are going to describe methodologies used to solve the Answer Type problem.

2.4.1 Augmentation-based Answer Type Classification of the SMART dataset

- ***Abstract***

  In this paper, The author solved the answer type classification task using data augmentation techniques. They used machine translation to popular languages, round-trip translation, and named entity annotation.

- ***Proposed solution***

  To further enrich its dataset and train its model's performance, several data augmentation strategies were implemented.:

  - **Dataset 1**
    Machine translation to **German**, **French**, **Spanish** and **Russian** is used for each question; In the end, we have five times more questions.
  - **Dataset 2**
    Round-trip translation ***English - German - English, English - Russian - English*** In the end, we have three times more questions.
  - **Dataset 3**
    Here the author annotates each entity in the question with an entity from the knowledge graph Dbpedia resources. Each named entity is replaced with one of its RDF types. These data are extracted using Spotlight. Overall, the dataset comprises 163,488 questions.

  Google Cloud Translation was used to automatically translate Dataset 1 and Dataset 2. Subsequently, all datasets were merged into a single dataset, referred to as Dataset 0.

  In this paper, the author developed a total of seven classifiers: Firstly, the category was classified, and then the corresponding model was selected based on the category. In the case of the category being "resource," the author developed five models to predict types: (type-1 ,type-2, type-3, type-4, type-5) Given that five types predicted, the list of types for the question is build and the last classifier is build to merge that types.

  NB: Each question in general has maximum of five types.

- ***Result Merging***
  To merge their results, each model produces a type as result R*ci*(Resource classification of type i) with $i \in \{0,1,2,3,4\}$. Each type has a weight. The merging of these types is computed while numerically calculating a weighted rank for each answer type that was predicted by at least one classifier pipeline. In their model, sf any of these classifiers predicts a literal or boolean response the outputs of other models are discarded.

- ***Experiments and results***
  They use BERT and BERT multi language cased models in their classification pipeline. the authors split the dataset into two parts; the first part is *train set* and the second part is *validation set*. The highest score on the test dataset was achieved with a merged combination of 3 predictions. We evaluated the weight combinations where each weight wi was chosen between 0:0 and 1:0 set, the sum of all used weights equals 1:0. The following best weight combination was created using this process: 30% D0, 30% D1 and 40% D1+MV. The fallback rank for the merging algorithm was taken equal to 10. This combination was submitted as the final solution for the task.

- ***Limits***
  The main limit of this work is that some terms may be confusing because they are translated. As they were using Round-trip-translation, some terms were replaced in the back translation for examples:

  - **Original**: Who Replaced Charles Evans Hughes as chief justice of The United States
  - **En-De-En**: Who Succeeded Charles Evans Hughes as chief justice of the United States
  - **En-Ru-En**: Who Replaced Charles Evans Hughes as chief justice of the United States

2.4.2 Semantic Answer Type Prediction using IAI at the ISWC task 2020.

- ***Abstract***

  In this paper, The author uses Transformers as BERT to Solve the Answer Type Problem.

- ***Proposed solution***

  The author proposed two approaches for category prediction

  - Feature-based classification: For category prediction, they use SVM classifier with word unigrams as feature and
  - Neural approach: They fine-tune a pre-trained BERT model (RoBERTa)

  For Type prediction, they proposed two approaches

  - IR-based methods
  - Neural Method

- ***implementation***
  They are using TF-IDF weighted word unigrams as features. Their implementation is based on*CountVectorizer* and *TFIDFVectorizer* class from the *sklearn* library with default parameters. They train an SVM Classifier with a linear kernel. The also implement a NAive BAyes classifier but they decided to exclude that after observing inferior performances.

- ***Method use to solve***

  - SVM: Support Vector Machine for category classification
  - BERT : RoBERTa for category classification
  - XBERT: X-transformers for type prediction
  - IR|IC: Type-centric IR approach for type prediction
  - IR/EC: Entity-centric IR approach for type prediction

  Result.
  The best accuracy is obtained by BERT model and it is **0.97**%

- *Limits*: The cost of deploying this solution is high.

2.4.3 The Combination of BERT and Data OverSampling for Answer Type Prediction

In this paper, the author combines the transformer model (BERT) with Data oversampling method. The used **Spacy v2.3.2** to analyse the question structure to get its semantic units, such as question type (what, who, when, ...), Subject, Main verb, Terms (Noun phrase, dependent noun phrases) in to build the sentence template.

The used Entity Linking method to match term to wikidata with the help of wikidata API. For Spacy, they take encore_web_lg, STOPWORD.

In Order to get sentence template, they ranked terms according to their length. for Example : Consider the following question:

- Question: "What kind of Film Does Ebenezer Kepombia Stars in ?"
- Category: "Resource"
- Type : ["Work","Film"]
- Question Template : "what kind of film, Ebenezer Kepombia, Stars
- Key-terms: ["Kind of film", "Ebenezer Kepombia", "Stars"],
- Subject: ["Ebenezer kepombia"],
- Main verb : ["Stars in],
- Entities: ["FIlm", "Ebenezer", "Ebenezer kepombia"],
- Question type: "What",
- Dependent nouns :["Film","Ebenezer Kepombia"]

For the **match function**, They consider the term which probability is higher than **0.9**. After that, they applied *oversampling technique* over linked terms to increase the number rare types . They replace the link terms with their labels, refer to the previous example we can produce some new questions. For the model, they decided to split the datatset into two part: D1: this dataset is used for category and type prediction for the boolean and the literal category D2: This one is used for resource type prediction. In total the build 4 BERT model for two dataset(two for DBpedia) and (two for wikidata) In D1 they have 5 categories : **boolean**, **literal-number**, **literal-string**,**literal-date** and **resource**, The author achieve the accuracy of **0.98%** on DbPedia and Wikidata.

While table 2.1 and table 2.2 showcase other approaches to address this issue, a common limitations of these methods are their significant resource-intensive implementation costs and their lack of flexibility and adaptability to changes in datasets, making them less effective in dynamic contexts.. In order to enrich the set of solutions to the Answer type problem and to address some of the limitations mentioned above, we propose **a knowledge graph refinement technique as a solution.**

2.5 Knowledge graph refinement techniques

There are various ways of building such knowledge graphs. They can be curated like *Cyc* [12], edited by the crowd like *Freebase* and *Wikidata*, or extracted from large-scale, semi-structured web knowledge bases such as *Wikipedia*, like *DBpedia* and *YAGO* [7]. Even though they are the most visible, this does not mean that they are perfect or even complete. Whichever approach is taken for constructing a knowledge graph, the result will never be perfect [7]. There is often a trade-off between coverage and accuracy, which is addressed differently in each knowledge graph. This is why knowledge graphs typically require improvements through graph refinement. What are the techniques for knowledge graph refinement? There are two main goals of knowledge graph refinement: (a) adding missing knowledge to the graph, i.e, **completion**, and (b) identifying wrong information in the graph, i.e, **error detection**. From a data quality perspective, those goals relate to the data quality dimensions free-of-error and completeness [7].

2.5.1 Error detection

Error detection is the process of identifying errors in a knowledge graph. It is important to identify errors in order to be able to correct them and improve the quality of the graph. Errors in a knowledge graph can be of different types, including:

- Data errors: These errors are due to incorrect or incomplete data.
- Logic errors: These errors are due to incorrect relationships between nodes.
- Structure errors: These errors are due to missing nodes or relationships.

There are many techniques for detecting errors in a knowledge graph, including:

- Manual inspection: This technique involves manually examining the knowledge graph to identify errors.
- Statistical analysis: This technique uses statistics to identify anomalies in the knowledge graph.
- Machine learning: This technique uses machine learning techniques to identify errors in the knowledge graph.

2.5.2 Kg completion

KG completion is the process of completing a knowledge graph by adding new data or modifying the structure of the graph. Kg completion is important for improving the coverage and accuracy of a knowledge graph. By adding new data, we can make the knowledge graph more completer and informative. By modifying the structure of the graph, we can improve the consistency and logic of the graph. There are many techniques for completing a knowledge graph, including:

- Data collection: This technique involves collecting new data to complete the knowledge graph.
- Data fusion: This technique involves merging data from different sources to complete the knowledge graph.
- Data generation: This technique involves generating new data from existing data to complete the knowledge graph.
- Machine learning: This technique uses machine learning techniques to complete the knowledge graph

The main difference between error detection and kg completion is that error detection aims to identify errors, while kg completion aims to complete the knowledge graph.

2.6 Conclusion

At the end of this section, the objective was to attain a profound comprehension of what an Answer Type is and provide its state of the art. It emerges that Answer Type is a crucial subtask within question-answering systems, significantly reducing the search space for answers by predicting the type of response to a question. This task relies on a semantically extensible data structure known as the knowledge graph. In terms of utility, a study has shown that users prefer knowledge graph-based search results over traditional search engines, hence its significance in domains such as Information Retrieval and user experience improvement. Knowledge graphs can also be employed in the fields of IoT and data analysis. The ability to reason over a knowledge graph makes it the most suitable structure for this type of problem. Regarding the tools used for addressing this problem, notable examples include RNNs, which are challenging to parallelize, and transformers such as BERT, T5, and GPT, which have demonstrated their performance.In the pursuit of improving Answer Type classification, several approaches, such as data augmentation and oversampling, combined with transformers, have emerged as effective strategies. Now, let's proceed to the methodology section, where we will outline the specific techniques and steps employed to tackle the Answer Type problem effectively.

System	Year	Accuracy	NDCG@5	NDCG@10
Hierarchical Contextualized Representation Models for Answer Type Prediction	2020	0.964	0.749	0.721
Reaching out for the Answer: Answer Type prediction	2021	0.991	0.734	0.658
Multilingual Hierarchical Expected Answer Type Classification using the SMART 2021 Dataset	2021	0.991	0.643	0.577
Semantic Answer Type Prediction by using BERT classifier and Rule-based Ranking Strategies	2021	0.985	0.737	0.702
CitySAT: a System for the Semantic Answer Type Prediction Task	2021	0.984	0.842	0.0854
The combination of BERT an Data Oversampling for Answer Type Prediction	2021	0.985	0.727	0.664
Semantic Answer Type Prediction using Dense Type Embeddings	2021	0.985	0.825	0.790
Semantic Answer Type Prediction using BERT* IAI at the ISWC SMART Task 2020	2020	0.977	0.804	0.793
Two-stage Semantic Answer Type Prediction for Question Answering using BERT and Class-Specificity Rewarding	2020	0.962	0.777	0.762
COALA - A Rule-Based Approach to Answer Type Prediction	2020	0.744	0.540	0.521
Semantic Answer Type Prediction using BERT* IAI at the ISWC SMART Task 2020	2020	0.97	0.68	

Table 2.1: Methods and evaluation for DBpedia

System	Year	Cat_match	mrr
Hierarchical Contextualized Representation Models for Answer Type Prediction	2020	0.96	0.59
Reaching out for the Answer: Answer Type prediction	2020	0.991	0.45
Multilingual Hierarchical Expected Answer Type Classification using the SMART 2021 Dataset	2020	0.980	0.430
The combination of BERT an Data Oversampling for Answer Type Prediction	2021	0.98	0.70

Table 2.2: Methods and evaluation for Wikidata

CHAPTER 3

APPROACH AND METHODOLOGY USED

3.1 Introduction

In this section, we propose our methodology for solving the Answer Type Prediction problem based on the KG refinement presented in the previous chapter.

3.2 Research methodology

Based on our experience in empirical research in software engineering, we design the research methodology. We consider that to solve Answer Type prediction research problem, SMART 2022 organizers provide two case study: the DBpedia and the Wikidata case studies. Case study research method is suited for understanding a complex real-world setting [13]. Thus, the aim of studying the cases of DBpedia and Wikidata KGs is to provide a deeper understanding of the AT problem so that the solution proposed can be generalize to any settings. On the other hand, we wanted to explore, test, evaluate (using as baseline the past challenges) different possible solutions that can be used to solve the AT prediction research problem and to propose the efficient ones. Thus, we completed our research methodology with action research. Action research is an iterative process in which plans are created, implemented, revised, then implemented, lending itself to an ongoing process of reflection and revision. During this process, findings emerged as interventions are implemented. It is used where major challenges cannot be studied without implementing them, and where implementation implies a long-term commitment because effect may take time to emerge. In addition, we were having the SMART organizers who describe the problem and wanted a solution to it. We set-up an action research method in which we as researchers and SMART 2022 organizers as the problem owner were engaged collaboratively to identify the problems and develop effective solutions. This solution is evaluated using the Test dataset provided by the SMART 2022 organizers. Given that at the end we proposed a solution based on software code, we used Scrum process [14].

The research methodology we applied consisted of a set of aggregated interventions to build the system. These interventions involved a series of actions taken during the development of the system in order to come up with an effective solution that can be used in any case. It's organized into the Pre-Intervention (Section 3.2.2), the Intervention (Section 3.2.3) and the Post-Intervention (Section 3.2.4). Before we present the different phases of our methodology, the research question is presented in Section 3.2.1.

3.2.1 Research question

Using the Answer Type prediction problem described by the SMART organizers [1], we defined the following research question: "For a question in natural language, what is the type of it answer?". To reply to this question, we should provide a system which takes as input a question and predict the type of the answer using a set of candidates from a target knowledge graph or an ontology. Given that two main types of answers can be given (question category and answer type), we derived the Answer Type research question into two specific questions:

- What is the category of a question in natural language?
- Given the category of a question, what is his type?

As presented in the introduction, the responses to these research questions are important to improve the performance of QA systems.

3.2.2 Pre-Intervention

The first task we did during the Pre-Intervention was to put in place a working group responsible to the development of the solution. Thereafter, a list of activities was designed and prioritized. These activities are:

1. Deep understanding of the Answer Type problem,
2. A review of existing solutions,
3. Identification of the solutions that can be implemented,
4. Definition of the list of activities that should be executed to solve the problem.

The list of activities that should be carry out (during the intervention phase) are organized using the Scrum Product Backlog. Thereafter, these activities are prioritized. These tasks are following:

1. Empirical analysis of the datasets,
2. Review of the previous challenges publications,
3. Definition of the model,
4. Training of the model,
5. Model refinement,
6. Model validation,
7. Model use on the test data.

3.2.3 Intervention

During the Intervention Phase, the solution to the problem is built, evaluate, refine, until we reach a certain value of evaluation. For instance, the statistical learning used in this paper is refine until we obtained the mean accuracy that we defined as our ground truth.

During the intervention phase, the list of interventions identified during the Pre-Intervention Phase are implemented. These interventions are executed in iterations. Each iteration is composed of the planning, the implementation, the validation and reflection.

1. **The planning:** consists of selecting a list of intervention and planning their execution. The planning is designed and enhanced iteratively. It is obtained from requirements, precedents implementations, problems encountered, etc. Once these elements are identified, the list of interventions is updated and the planning is updated.
2. **The implementation:** consists of the development of the solution.
3. **The validation:** consists of testing the model on the test data and calculate the accuracy. During the implementation and the validation of the solution, new learning may emerge.
4. **The reflection:** consists of drawing lessons, revise and prioritize the list of interventions.

The iteration nature of the process provides us opportunities for improvements to the knowledge, methods, and better understanding of the approaches and methods.

3.2.4 Post-Intervention

The Post-Intervention consists of assessing the model using the test data furnished by the SMART 2022 organizers. The source code [1] is also published using Open Source licence and is available on Github.

3.3 Dataset Description

In this part we were learning the structure of the dataset to provide a profound or deeper comprehension of it such as present a good solution. For this work, we are using the SMART-2022 dataset. It, owned by the International Semantic Web Conference, is a valuable resource for the research community in the field of Semantic Web. It consists of a collection of data specifically curated and prepared for the conference. ISWC is a prominent event that brings together researchers, practitioners, and experts from around the world to discuss the latest developments and advancements in semantic technologies. This dataset is composed of a set of question which can be generally classify on **Wh-term** (Who, What, When, Where, Which, Whom, Whose, Why). and **none Wh-term** (Do, At, Is, How, The, To, Tell, In ...) these question are represented in a Json file. Each question have this format :


```
{
    "id":1,
    "question":"What is the name of the opera based on Twelfth Night ?",
    "category":"resource",
    "type":[
        "dbo:Opera",
```


[1] Link: https://github.com/smart-task/smart-2022-datasets

```
        "dbo:MusicalWork",
        "dbo:Work"
    ]
},


{
    "id":3,
    "question":"Do Prince Harry and Prince William have the same parents?",
    "category":"boolean",
    "type":[
        "boolean"
    ]
}
```


[2]

As presented by the organizer's datasets are designed to provide a single answer category either "boolean", "literal", or "resource". They assign the "boolean" category as boolean, "literal" category into an answer type either "number", "date", or "string". For resource categories, DBpedia and Wikidata ontologies classes are used to identify the answer type. To understand the dataset, we thought it necessary to investigate a subset of this dataset. Thus, fifty questions for each dataset were randomly selected and their annotations automatically removed. Once the annotations were removed, a manual annotation of this subset of the dataset took place. The manual analysis of the dataset gave us the insight that "resources", "literal" and "booleans" are not equally distributed in the dataset. The analysis of the distribution by category is presented by the Fig. 3.1

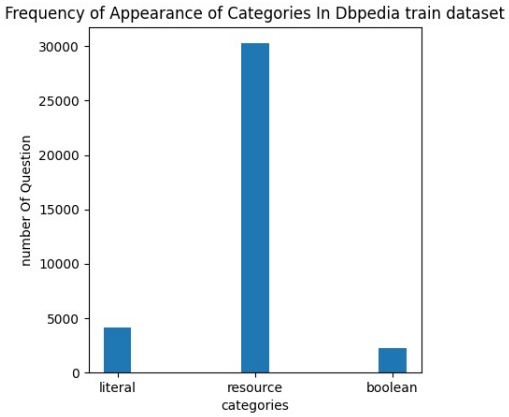


Figure 3.1: Inegal repartition of category in DbPedia dataset

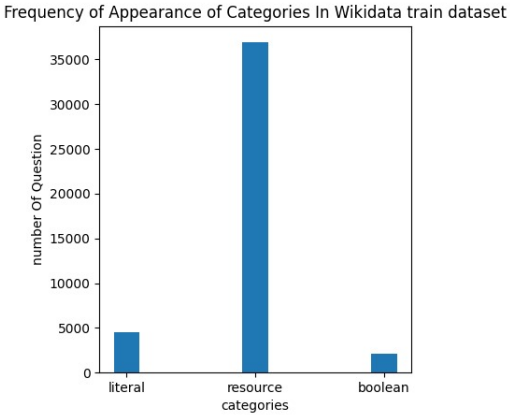


Figure 3.2: Inegal repartition of category in wikidata dataset

for DBpedia and Fig. 3.2 for Wikidata confirmed this insight. The main finding in this dataset was that:

- Data are not equally distributed amongst the different categories (see Fig. 3.1 for DBpedia and Fig. 3.2 for Wikidata);
- The question can be divided into Wh-terms (What, When, Which, Where ...), and none wh-terms (How many, Off which, Thus, Did, etc). questions.
- Some questions were noisy see tab 4.5

[2] this is only two examples of question in our dataset

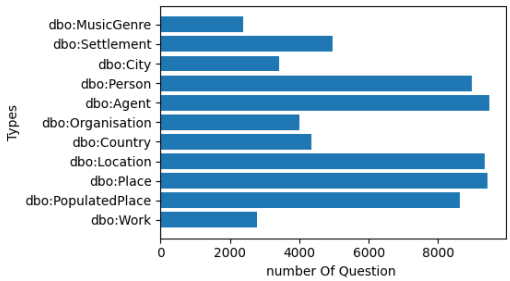


Figure 3.3: Commonly occurring Resource Types in DbPedia

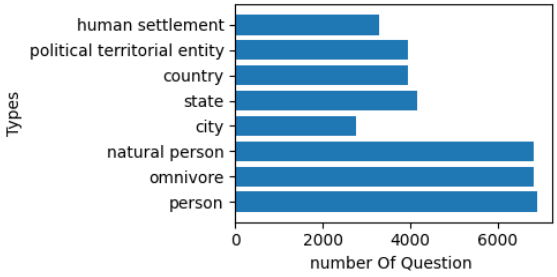


Figure 3.4: Commonly occurring Resource Types in Wikidata

Wikidata ontology has around 50K classes, while Dbpedia has around 760 classes. Figures 3.4 and 3.3 represent the most frequent types in our dataset. Based on the insights obtained from the dataset analysis, we create the question taxonomy as depicted in Figure 3.8. This hierarchical structure will guide the construction and enhancement of the model.

3.4 Model Definition

The objective of the model definition stage was to establish a model capable of addressing the Answer Type prediction research problem across various datasets. To begin, we formulated a graph structure that can effectively represent any Question Answering dataset 3.5. This involved aligning the taxonomy from Figure 3.8 with the corresponding answer types and categories. Subsequently, we matched the set of questions with the question taxonomy. The idea behind this approach is to establish that if a question is connected to a node within the taxonomy, and that node is associated with a category and/or question type, then the question is also linked to that particular category and/or answer type. To establish this relationship, we introduced the *"hasEmbedding"* relation between a question and its embedding in the taxonomy. To define this relation, knowledge should be extracted from the question. Knowledge extraction is the creation of knowledge from structured (relational databases, XML), semi-structured (source code) and unstructured (text, documents, images) sources [15]. In our case, we are faced with extracting terms from unstructured sources which are questions. From the Fig. 3.5, we define the Question Answering dataset as a multi-relation data using a Knowledge graph defined by the quadruple $G = (V, E, R, T)$ where:

- $V = \{v_i \in V\}$ the set of nodes. These nodes are questions, questions categories, and questions types;
- $E = \{v_i, r, v_j\}$ the set of edges, which are triples with node v_i connected to node v_j via the relation r. These are the relations between two questions, a question and his type or his category;
- $T = \{t(v_i)\}$ denotes the nodes types. The type of nodes categories and nodes types are their labels, and the types of node questions are their id and the title of the question;
- $R = \{r \in R\}$ denotes the relation types. These are the labels of the relations.

- ***hasEmbedding:*** this relation defines how closeness a question is with an embedding vector defined in our taxonomy of question (see Fig. 3.6). This is very useful because when two questions have the same embedding in the taxonomy, they will have the same category and the same type.
- ***isCloseTo:*** this defines a relation between two questions that have the same embedding. This information will be particularly important during the training of the model. In effect, two questions that are close to each other share the same category and the same type.
- ***hasCategory:*** this relation defines the relation between a node (the node can be a question or an embedding vector in the taxonomy) and its category;

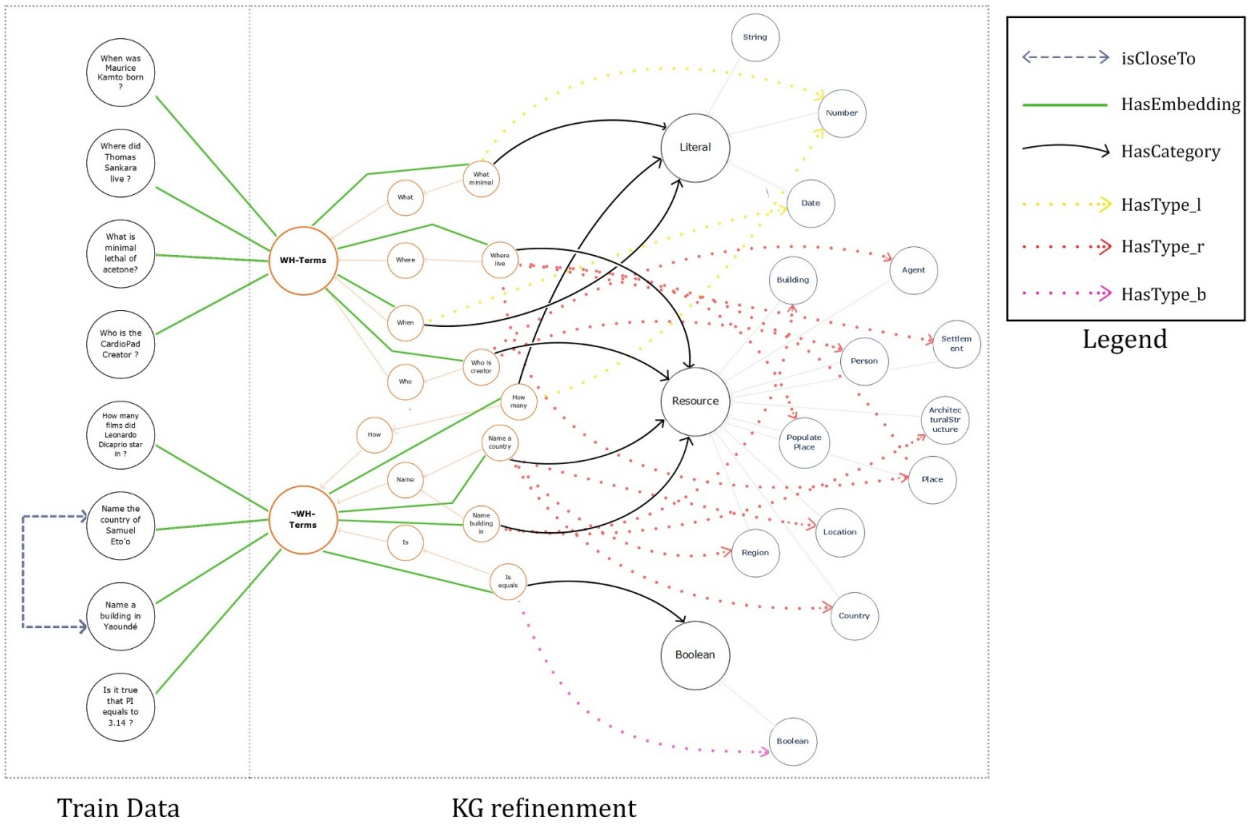


Figure 3.5: Knowledge graph-based representation of the dataset

- ***hasType_b:*** this relation is used to model the relation of "Boolean" category to its type, which is "boolean";
- ***hasType_l:*** this relation is used to model the relation of "Literal" category to its type, which can be "number", "string" or "date";
- ***hasType_r:*** this relation is used to model the relation of "Resource" category to its type in the knowledge graph.

Théorème 3.4.1 *Given two questions that are $Q1$ and $Q2$. If $Q1$ and $Q2$ shares the same embedding vector in the taxonomy, thus, the triple (Q1, isCloseTo, Q2) holds and $Q1$ and $Q2$ has the same category and the same type.*

Given this new configuration of the training dataset, we reduced the problem of Answer Type prediction to the problem of KG completion. To solve this problem, our goal will be to provide an efficient tool to complete the graph by adding new facts without requiring extra knowledge.

Inspired by the TransE algorithm [16] for learning low-dimensional embedding of entities, we define our model, presented by the equation 3.1. We consider that relationships are represented as translations. For instance, if (Q1 hasSameEncoding Q2) holds, then the embedding of $Q1$ should be close to the embedding of $Q2$. The model is defined as a function which takes the node and a relation and predict all the nodes that are related to this relation. This prediction is based on the proximity between two nodes.

$$\begin{aligned} f_r : & V \times E \times V \longrightarrow \mathbb{R} \\ & f_r(v_i, r, v_j) = ||v_i + v_r - v_j|| \\ & v_i + v_r \simeq v_j \text{ if } f_r(v_i, r, v_j) < \beta \end{aligned} \tag{3.1}$$

The function f_r is used to verify if the triples in the knowledge graph holds. The triple (v_i, r, v_j) holds if $f_r(v_i, r, v_j) < \beta$ - in this case, we can say that $v_i + v_r \approx v_j$. β being the model parameter. If β reaches a certain value (or belongs to a certain interval of values), thus, we can say that v_i and v_j are linked with the relation r.

The generic model that we just defined is based on node embedding, the nodes being the questions, the questions categories and the questions types. Thus, we should find a way to model these nodes in such a way that they can be used to train our models to predict relations. This task is done by encoding the questions so as to simplify the definition of (Question, hasEmbedding, Vector) triples. We extracted knowledge from the question title and used this knowledge to define the embedding vector of each question. To this end, we suppose that a question can be well represented by some terms (word or group of words) in his title and we defined the $question2vect$ function (see equation 3.2) which takes a question and return a vector containing the terms used as the embedding of this question.

$$question2vect(Q) = \{w_1, w_2, ..., w_n\} \tag{3.2}$$

In equation 3.2, $(w_i, i <= n)$ denotes the different terms (word or group of words) in the question title that can be representative of the question. For instance, for category prediction, these words can be the "WH-", "Is", "How many" terms that are used to identify the category of a question.

For instance, using the taxonomy of fig. 3.7, the size of the vectors returned by $question2vect$ is 15. Using this taxonomy, we provide by the listing 3.1 some examples of the evaluation of $question2vect$ function using some questions in the dataset. The listing 3.2 presents the algorithm executed to transform a question into a vector of terms.

Listing 3.1: Example of the transformation of some questions into a vector using the taxonomy of questions

```
question2vect(what is branch of biology that starts with z ?)=
    ( What, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 )
question2vect(Who are the gymnasts coached by Amanda Reddin ? ) =
    ( 0, 0, 0, 0, Who, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 )
question2vect(How many superpowers does wonder woman have ?) =
    ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, How many, 0, 0, 0 )
```

Listing 3.2: Question2vect algorithm

```
Algorithm_Question2Vect
Input :
    Q // The set of questions
    T // The current taxonomy
Output :
    V // The vector of terms
Steps :
1   listOfWord = tokenize(Q) // Tokenize the question to obtain
                             the list of words it contains
                             Complexity = O(|Q|) where |Q| is
                             the number of question
2   for w in listOfWord
        if w in lessFrequent
            delete w    //Complexity = O(|listOfWord| * |lessFrequent|)
                             where |listOfWord| is the number of word
                             in the question and |lessFrequent| is
                             the  number of less frequent words
3   V = buildVector(listOfWord , T) // This method will
                    build the vector by replacingthe empty space by 0
                    so that the size of the vector correspondto |T| .
                    Complexity = O(|listOfWord| * |T|) where |T| is the
                    size of the taxonomy
```

3.5 Models Training and use

Before the models are trained, the first thing we did was to encode the questions so as to simplify the definition of (Question, hasEmbedding, Question) triples. We extracted knowledge from the question title, and use this knowledge to define the embedding vector of each question. To this end, we suppose that a question can be well represented by some words in his title and we define the $question2vect$ function (see equation 3.2 which takes a question and return a vector containing the terms used as the embedding of this question.

- **Learning the "hasEmbedding" relation:** the function $f_{hasEmbedding}$ is given by the equation 3.3.

$$f_{hasEmbedding}(Q, hasEmbedding, V) = |Q - V| = \alpha \tag{3.3}$$

Listing 3.3: HasEmbedding algorithm

```
Algorithm_HasEmbedding
Input :
    Q // The set of questions
    V // The current taxonomy
Output :
    hasEmbedding = false //A boolean defined if the question
                         Q has Embedding V
Steps :
1   W = question2vect(Q)
2   if ||W-V|| < beta
        hasEmbedding = true
3   else
        hasEmbedding = false   .// Complexity = Complexity
                                   question2vect
```

- ***Learning the "isCloseTo" relation:*** the function $f_{isCloseTo}$ is given by the equation 3.4.

$$f_{isCloseTo}(Q_i, isCloseTo, Q_j) = |Q_i - Q_j| = \alpha \tag{3.4}$$

Listing 3.4: isClosedTo algorithm

```
Algorithm_isClosedTo
Input :
    Q_1, Q_2 : question// Two questions
Output :
    isClosedTo  = false //A boolean defined if the question
                        Q_1 is closed to Q_2 or not
Steps :
1   theta = 0
2   Remove the special characters inside Q_1 and Q_2 \,/"#:%
3   Put all the word of Q_1 and Q_2 in lower case
4   Tokenization of Q_1 and Tokenization of Q_2
5   For each token t_i of  Q_1
        if t_i is inside the Tokens List of Q_2
            theta = theta + 1
        if theta < gamma
            isClosedTo = true
Complexity O(|Tokenize(|Q_1| * |Tokenize(|Q_2|)
```

Globally, two vectors are closed if $\alpha \approx \beta$, β being the model parameter define during the training process.

- ***Learning "hasCategory":*** to predict "hasCategory", relation, we consider that each embedding node has a probability to belongs to a category. The scoring function defining this probability can thus be obtained

using statistic learning, Naïve Bayes, etc. For instance, if we consider that we are using statistic learning, the scoring function will be obtained by the equation 3.5.

$$f_{hasCategory}(V, hasCategory, C) = \frac{\sum_{i=1}^{N} freq((V, C), Train)}{\sum_{i=1}^{N} freq((V, *), Train} \tag{3.5}$$

Listing 3.5: hasCategory algorithm

```
Algorithm_hasCategory
Input :
    V  // A vector
    C // The set of category
Output :
    c_v  = // The category of the vector
Steps :
1    For each c_i in C
2        nb = number of times V has category c_i in the train
              dataset
3        N = number of times V has any category in the train
             dataset
4        beta_i = nb/n // The probability that V has category ci
5        if max(beta_i) // Verify if beta_i is the max
                              probability obtained
6         c_v = c_i

Complexity O(|C|)
```

- ***Learning "hasType_b", "hasType_l" relations:*** these relations are predicted using simple rules. In effect, once the category is predicted, we defined a set of rules that match each element of the taxonomy to the corresponding type.

Listing 3.6: hasType_b algorithm

```
Algorithm_hasType_b
Input :
    V  // A vector
Output :
    boolcat  = // A boolean to know if the category is

Steps :
1    if question.category= "Boolean":
2        boolCat = Boolean

Complexity O(1)
```

Listing 3.7: hasType_l algorithm

```
Algorithm_hasType_l
Input :
    V  // Embedded vector
    C // category of question
    literal // List of Literals
Output :
    t  = // Type of embedded vector
Steps :
```

```
1   if category(C,V) = Literal: // check if the vector has
                                    literal as category
//Search for the litteral  in the list of literals that
// match with the vector V
2       t = matchLiteral(literals , V)
Complexity O(1)
```

- ***Learning "hasType_r" relation:*** concerning the $hasType\_r$ relation, we combine the internal method for knowledge graph refinement with the external method. In effect, we define the embedding vector of the question, so that if it match to a category provided by the SMART organizers, thus, this category is saved. If not, we made a SPARQL query on the graph online in order to get the category that holds.

Listing 3.8: hasType_r algorithm

```
Algorithm_hasType_r
Input:
V : Vector // a Embedding vector of a
T : Taxonomy // The taxonomy
Output :
R : List <type> // List of type of the KG
Steps :
1  Search inside the taxonomy T , which embedding vector has the same
valueof V
2   If the vector V found in step {1} has one or multiples type value
de fineinside our KG , insert them inside R List
3   Else
3.1    define W = ToString(V)
3.2    Search on the external KG using Wikibase API
3.3   With the result of step {3.2} , execute SparQL query to get
the types associated
3.4   insert the types found on {3.2} inside R
4 ) Return R
/ / The ToString(V) function transforms return a String
            by concatenating each term o f V
```

Listing 3.9: SPARQL query over Wikidata for hasType_r

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbr :<http://dbpedia.org/resource>
PREFIX dbo :<http://dbpedia.org/ ontology >
WHERE{
'+Search_parameter+' rdf:type ?type

FILTER strstarts(str(?type), str(dbo:))
}
//comment: Search_parameter must be replace by the entities found on
step{3.1} of hasType_r Algo
```

Listing 3.10: SPARQL query over Wikidata for hasType_r

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX mwapi: <https://www.mediawiki.org/ontology/API#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>

SELECT DISTINCT ?type ?typeLabel
WHERE {
```

```
    wd:'+Search_parameter+' (wdt:P279 | wdt:P31) ?type .
    SERVICE wikibase:label {
      bd:serviceParam wikibase:language "en" .
    }
}
ORDER BY ASC (?type) LIMIT 5
//comment: Search_parameter must be replace by the entities found on
step{3.1} of hasType_r Algo
```

Implementation To extract knowledge from questions, we used TF-IDF technique. We identify the less frequent words in questions and the stop words and we removed so to obtain the information that can be used to well represent the question. These informations were used to define the $f_{hasEmbedding}$ and $f_{isCloseTo}$ models. Thereafter, we define a function which calculates the distance between two questions. This consist of subtracting the terms that are identical and count the rest of the terms in the result. If the rest is less than 2, thus, the two nodes are close. To train $f_{hasCategory}$ model, we used the Python programming language. Thereafter, DBpedia and Wikidata provided were divided into training (80%) and testing (20%) (Fig. 3.6 illustrate the dataset). The 20% of the dataset was selected proportionally to the distribution of each data category. Tables 3.2 and 3.1 gives the details on the training and the validation datasets, both for DBpedia and Wikidata. From the 80% of the training dataset, we removed all the relations that were provided as response to the Answer Type prediction question. In this figure 3.6, the dot lines represent links that are predicted

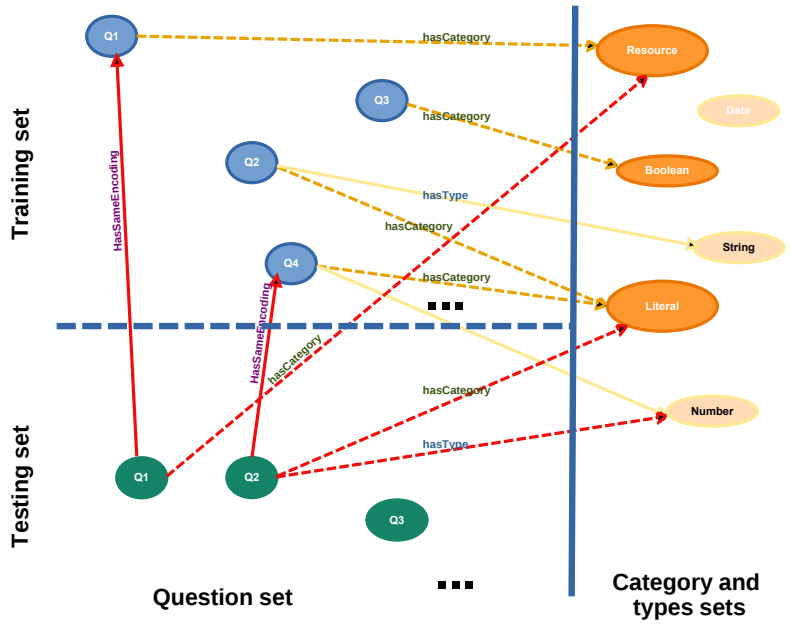


Figure 3.6: Representation of the training and the testing datasets.

	Wikidata				DBpedia			
Category	Resource	Literal	Boolean	Total	Resource	Literal	Boolean	Total
Training set	36886	4530	2138	43554	30226	4217	2227	36670
Validation set	???	???	???	10855	???	???	???	9104

Table 3.1: Tabular representation of the SMART 2022 training and testing dataset

Using the training datasets, we compute the function $f_{hasCategory}$ (see equation 3.5) for each category. This is done by calculating the frequence of apparition of a node embedding V classified as for the category C divided

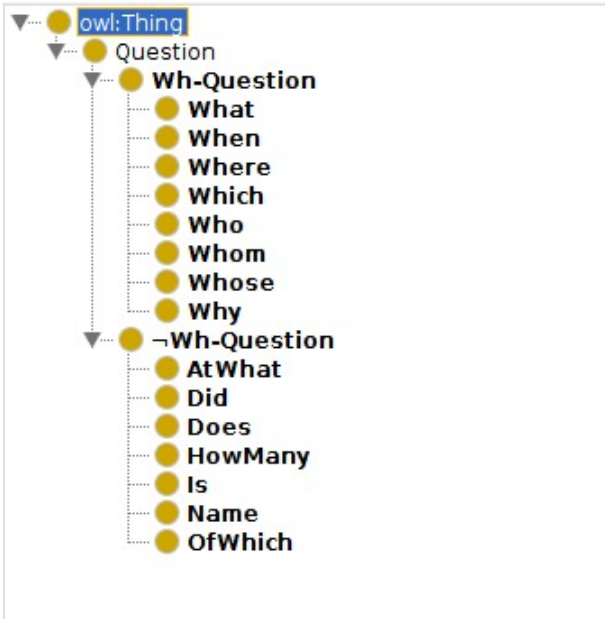


Figure 3.7: First taxonomy

	Wikidata				DBpedia			
Category	Resource	Literal	Boolean	Total	Resource	Literal	Boolean	Total
Training set	29508	3624	1710	34842	24180	3373	1781	29334
Validation set	7377	906	427	8710	6045	843	445	7333

Table 3.2: Tabular representation of the 80% of train and 20% of validation repartition of the dataset used to train our model

by the number of apparition of the node embedding V with any category. Once trained on the training dataset, we applied it to the test dataset and obtained the accuray of 0.89 for DBpedia and 0.84 for Wikidata. These values were very low compared to the ground truth that we fixed during the Pre-Intervention. Thus, we decided to refine the model. This is presented in the next section.

3.6 Model refinement and validation

After a deep reflection during ad hoc meetings between the researchers, it was decided to add a new intervention consisting of updating the taxonomy of the Fig. 3.7 and to test other models. This new intervention was put in the main priority and two more developers having good knowledge in Naive Bayes classifier joined the group. The second Sprint was dedicated to the development of the taxonomy of questions. The taxonomy of question was extended to obtain the more larger number of nodes (see Fig. 3.8). The statistical scoring function and the Bayes function was applied to the training dataset and we obtained the best accuracy of 0.95% on DBpedia and 0.93% on Wikidata.

3.7 Conclusion

In this section, we were supposed to propose our approach for solving the AT problem. It turns out that we produced a graph structure that allows us to represent all Question Answering datasets, and that this structure is

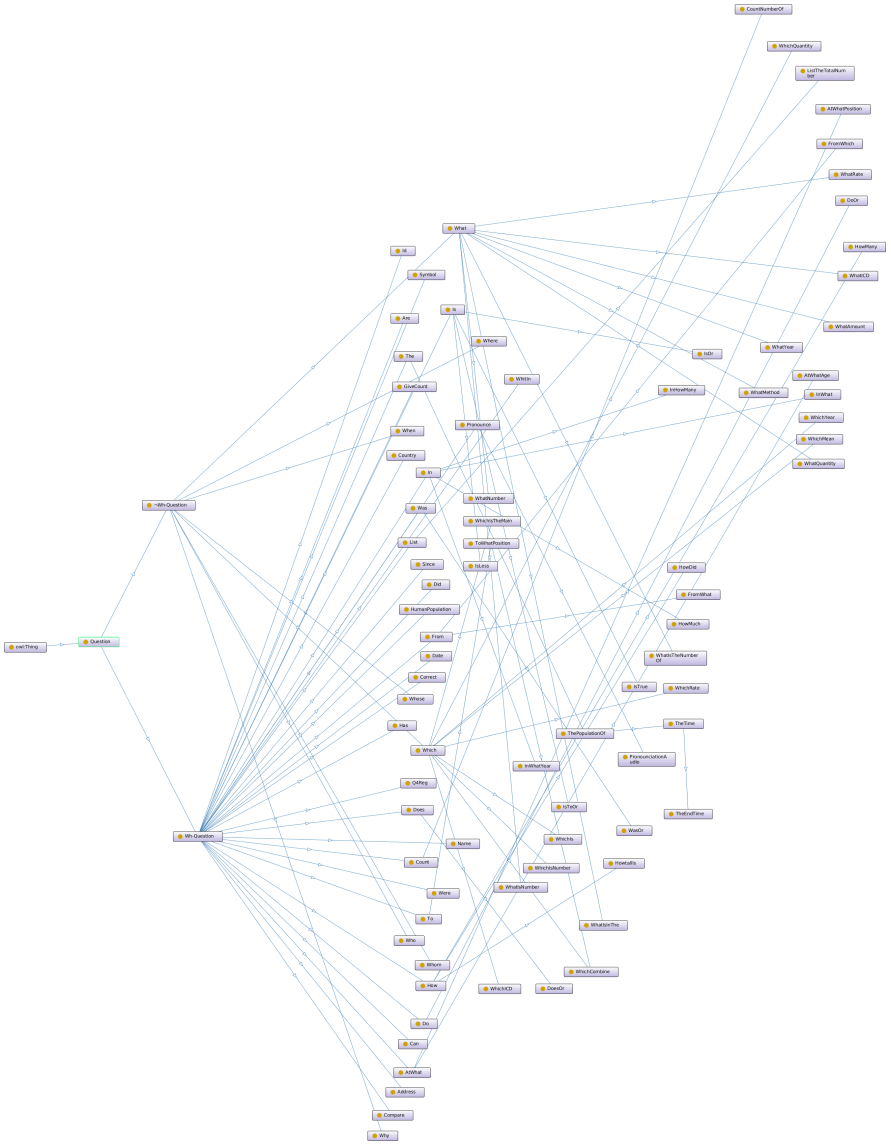


Figure 3.8: Final Taxonomy

called a taxonomy. Then we defined our model and trained it. The intervention phase allowed us to obtain a model with the accuracy superior to the ground truth. Thus, to finalize our work, we applied our approach on the test data provided by SMART 2022 organizers. And we got our results analyzed by the SMART organizers. The result of the different models is presented by the Table 4.1 and 4.2.

CHAPTER 4

EXPERIMENTATION AND RESULTS

4.1 Introduction

In this section, we will explain how we proceeded to solve the AT problem, the different steps of resolution and especially the results obtained after evaluation by the challenge organizers.

4.2 General Idea

Since the type of a resource is available from the KG ontology and the answer to a question is unknown at the outset, our first idea was to be able to produce a vector/element from a question from which we can query the online knowledge graph and then find the associated type. Similarly, we have found that it is quite possible to classify questions and associate them with one or more resource types. Given that classification seems to us to be less computationally demanding (queries, execution time) than making a series of SPARQL queries on KGs and that classification seems to us to be a priori less complex than producing a vector that will be associated with the type of the resource in order to search online on the KG. We have chosen to:

1. Start by producing a simple classification of some questions and associating them with corresponding resource types.
2. Then search online for the elements that we have not been able to integrate into our classification for their type.
3. Evaluate the overall results and refine our classification in order to make as few SPARQL queries as possible.
4. Repeat this process until satisfaction

4.3 Experimentation

4.3.1 First Experimentation

In this first experimentation, we focused our attention only on resource type, we extracted all questions which have resource as category from the dataset. For example, in DbPedia dataset we obtain 30.266 resources on 36.670 questions is around **80%** of the main dataset.

with the aim of establishing a first classification of resource according to questions:

- - We have firstly identified frequently resource type in our dataset with their association for example:
  - + We have 11.970 occurrences for the type "dbo:Place" over 30.266 resources in our Train set. This type will generally be associated to types "dbo:Location" with an occurrence of 11.831, the type "dbo:PopulatedPlace" with an occurrence of 11.005 ...
  - + We have 10.111 occurrences for the type "dbo:Agent" for 30.266 resources in our train dataset arround 33.45%. This same type will generally be associated to "dbo:Person" 9.824 with occurrences in our train. ...
- - We have also identified the most frequent word which represent all the times these types that we have identified.
- - We finally establish a first classification of these frequent type with the term that we have selected. For example: All question which start by a Wh-Term, starting by **where is** and containt one the terms **"born"** or **"birthplace"** had the types "dbo:Place" and "dbo:Location". regex : ("where is).*(birthplace|born) .

We have therefore initially established a fairly simple classification of frequent terms.

Then for a question, we build a **vector** from which we find the type of the resource link to the question in the knowledge graph. For example for the question :"question": "where was shotton born ?", we find differents entities linked to this question . By knowing the name of entity *Shotton* and then we get the following types:

```
[
    dbo:Person
    dbo:Athlete
    dbo:SoccerPlayer
    dbo:Artist
    dbo:Cricketer
    dbo:BaseballPlayer
]
```

The answer contained in the train being :

```
{
    "id": 42112,
    "question": "where was shotton born",
    "category": "resource",
    "type": [
      "dbo:Person",
      "dbo:MilitaryPerson",
      "dbo:Agent"
    ]
}
```

It quickly became clear to us that a question can have an ambiguity problem because in fact there are several people with the name Shotton. In addition, the type associated with this question ("Where was shotton born ?") seemed to us to be

```
["dbo:Place","dbo:Location","dbo:Country", "dbo:PopulattePlace"]
```

and not

```
["dbo:Person","dbo:MilitaryPerson","dbo:Agent"].
```

This observation therefore led us to deduce that there are errors in the training data and we opted to prioritize the type classification with a process that we followed step by step manually in order to maximize accuracy.

4.3.2 Second Experimentation

With the objective of refining our classification as much as possible, we have continued to study the frequencies of appearance of the types with a question (a few terms as well as their orderings in the question). We then decided that given a question, we are going to build a vector to do our classification, this vector translates a regex that we can apply to the question. So, we started from what we called Interrogative-Terms:

```
[
    "which", "what", "whose","when", "who",
    "whom", "where", "how", "why","whether",
    "whatsoever", whither", "whence", "whatever", "wherever"
]
```

We considered only the questions which include only one Interrogative-Terms in order to bring out a classification for the types. For example: "In what county santa maria is ?", "Where was borys szyc born?", "what kind of book is on photography?","where is the birthplace of joseph proust",...

From the frequencies of appearance and the ordering of terms in a question, we have carried out a regex-based question classification. This allowed us to deduce that:

- - a question starting with "which" and followed by "is the name of" and ending with "business" respectively, generally has a high probability of having the resource type either ["dbo:Company", "dbo:Organization" , "dbo:Agent"]. regex: ("which).*(is the name of).*(business)
- - a question starting with "what film" usually has the types ["dbo:Film","dbo:Work"] associated. regex: ("what movie).*
- - a question starting with "what type of" followed by "celestial object" has the type "dbo:CelestialBody". regex: ("what type of).*(celestial object).*

After having assigned the types to the questions based on our classification, we also had to carry out an automatic filtering in order to remove the duplicates or even delete a type that we suspect is incorrectly assigned with a criterion of divergence/disjointness between the types . For example, the type "dbo:Person" and "dbo:Place" cannot both be in the answer of a question.

For the types that we have not yet been able to integrate into the classification, we searched for them using the SPAQL query as presented in the first experiment.

4.4 Pipeline

the following pipeline better presents the process followed:

- 1. Get Question which have resource as category from the Training dataset.
- 2. noted the frequencies of appearance of the types according to terms and the ordering of terms in the question.
- 3. integrate the type and the regex representing the frequent terms and their ordering in our classification if the frequency of appearance of the type for these identified terms is high (> 0.7)
- 4.integrate the following questions into the classification taking into account the strongest frequencies such as our sheets.
- 5. Go through the data test questions one by one in our classification to assign the types
- 6.Filter out duplicates and disjoint/divergent types if any
- 7. Execute the research on the KG if we don't have a type associated to this question with our classification
- 8. Save the result in a JSON file

Scoring function	Accuracy	Approach for resources type	NDCG@5	NDCG@10
Statistic WH (taxonomy = 15 classes)	0.94	Statistic WH (taxonomy = 15 classes)+ KG matching	0.346	0.303
Statistic WH (taxonomy = 15 classes + other resources)	0.954	Statistic WH (taxonomy = 15 classes + other resources) + KG matching	0.357	0.400
Statistic WH (taxonomy = 50 classes + other resources)	0.964	Statistic WH (taxonomy = 50 classes + other resources) + KG matching	0.320	0.393
Statistic WH (taxonomy = 67 classes + other resources)	0.957	Statistic WH (taxonomy = 67 classes + other resources) + KG matching	0.392	0.435
Bayes + KG matching	0.954	Bayes + KG matching	0.389	0.432

Table 4.1: Results of the answer type prediction on DBpedia test data

Accuracy	MRR (KG Matching approach)
0.94	0.15

Table 4.2: Results of the answer type prediction on Wikidata test data

4.5 Result

In this chapter, we present the results of our study on the prediction of the type of answer to questions. It is important to remember that we proceeded with a knowledge graph refinement method. We found that our method is particularly effective for predicting the types of answers. In addition, our approach is applicable to the vast majority of datasets in Answer Type in particular and Question Answering in general. The evaluation of our solution was carried out by ourselves, the results are included in the table 4.1 and table 4.2 and the final evaluation was carried out by the organizers of the SMART TASK challenge and these results are listed in the table 4.3. Our code is available on this link our source code [1]

[1]Link: https://github.com/Zepe237/SMART-3.0/tree/master

System	Year	Accuracy		NDCG@5	NDCG@10	MRR
		DbPedia	Wikidata			
COALA - A Rule-Based Approach to Answer Type Prediction	2020	0.744		0.540	0.521	
Two-stage Semantic Answer Type Prediction for Question Answering using BERT and Class-Specificity Rewarding	2020	0.962		0.777	0.762	
Hierarchical Contextualized Representation Models for Answer Type Prediction	2020	0.964	0.96	0.749	0.721	0.59
Towards an Approach based on Knowledge Graph Refinement for Answer Type prediction	2022	0.964	0.94	0.435	0.392	0.15
Reaching out for the Answer: Answer Type prediction	2021	0.991	0.99.1	0.734	0.658	0.45

Table 4.3: Results of DbPedia and Wikidata Answer Type systems

4.5.1 Evaluation Metrics Description

Every work should be evaluated in order to be compared to existing solutions. Here we are going to Describe the Metrics use to evaluate the Answer Type.

Accuracy

Accuracy is a machine learning metric that is used to evaluate or quantify the performance of a model. It measures the proportions of correct predictions made by the model. Accuracy is calculated using a confusion matrix. The confusion matrix summarizes the results of the model's predictions. This matrix is represented by the table 4.4 where

- TP (True Positive) : The system correctly predicted that the instance was positive
- FP (False Positive) : The system incorrectly predicted that the instance was positive.
- TN (True Negative) : The system correctly predicted that the instance was Negative
- FN (False Negative) : The system incorrectly predicted that the instance was Negative

. Therefore, accuracy is calculated using the formula 4.1

	P	N
T	TP	TN
F	FP	FN

Table 4.4: Confusion Matrice

$$accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{4.1}$$

. In our case, this metric is used to evaluate the performance of category prediction by dividing the number of true categories predicted by our model per the total number of questions.

NDCG@5 & NDCG@10

The Normalized Discounted Cumulative Gain (NDCG) is a widely used evaluation metric for learning-to-rank (LTR) systems. NDCG is designed for ranking tasks with more than one relevance levels [17]. This metric takes into account the relevance of the results and their ranking order. ML teams often use NDCG to evaluate the performance of a search engine, recommendation, or other information retrieval system. The NDCG@5 and NDCG@10 notation is a variant of NDCG that focuses on the top five and ten results, respectively.

$$NDCG = DCG(ranked_list)/DCG(ideal_ranked_list) \tag{4.2}$$

$$DCG = \sum_{i=1}^{m} \frac{gain(i)}{log2(i+1)} \tag{4.3}$$

- **gain(i)**:is the relevance of the i-th result [18]
- **m**: is the total number of results

NDCG is calculated using the following formula 4.2 & 4.3 And it measures the quality of a ranked list. In our case, it was used to evaluate the order in which **response types** were returned. This order was defined by the challenge organizers. This metric was only used in the DbPedia case. For Wikidata we were using **MRR**.

MRR

MRR also allows for the assessment of the quality of a ranking, especially the relevance of the first results. It is not very sensitive to variations in the size of datasets. It's calculated by using the formula 4.4

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{Q} \frac{1}{rank(i)} \tag{4.4}$$

- Q is the set of queries that are being ranked.
- rank(i) is the rank of the first relevant document for query i.
- |Q| is the number of queries in the set.

4.6 Limits

At the end of this work, we can highlight several limitations.

- - Indeed, unfortunately, we could not complete our classification. Indeed, it takes around 53% of the training set. This therefore means that there is a great margin of improvement for this approach.
- - The training data available to us sometimes presented, but very rarely, results that seemed doubtful to us, such as:

```
{
    "id": 42112,
    "question": "where was shotton born",
```

```
    "category": "resource",
    "type": [
      "dbo:Person",
      "dbo:MilitaryPerson",
      "dbo:Agent"
    ]
}
```

It therefore seems important to us to establish our classification with more appropriate training data in this case.

- If the question has two parts, the model will not be able to effectively produce a response. (We have not had this kind of question in our dataset.)
- - The presence of duplicates in the dataset could compromise the quality of the results.
- There is too much noise in our dataset which can reduce the performance of our system. see table 4.5
- - The SPARQL query-based search approach was generally not successful from what we observed.

Question	id	Category	Types
Name an actor.	27835	resource	dbo:Person, dbo:Agent
Name an actor.	28055	resource	dbo:Person, dbo:Agent
Name an actor.	31185	resource	dbo:Person
Name an actor.	43681	resource	dbo:Person, dbo:Writer
Name an actor.	44832	resource	dbo:MusicalArtist
What is it?	116	resource	dbo:Person, dbo:Agent
What is it?	4120	resource	dbo:EthnicGroup
What is it?	6342	resource	dbo:Taxon, dbo:TopicalConcept
What is it?	6357	resource	dbo:City, dbo:Settlement, dbo:Populated Place, dbo:Place, dbo:Location
What is it?	16164	resource	dbo:Person, dbo:Agent

Table 4.5: Noise Contained in Dataset

4.7 Discussion

Based on the results published by the challenge organizers, we found that our method achieved performance that is consistent with the literature. It surpasses most methods on some points, such as execution time, implementation cost and performance. A crucial point to emphasize is the exceptional flexibility and adaptability of our approach compared to other systems. In comparison, our method demonstrates a remarkable ability to adjust to changes of the datasets, thereby highlighting its significant advantage in dynamic contexts. The best results are obtained by line 5 of the table 4.3 and it uses pre-trained language models such as Roberta (Robustly optimized BERT Pre-training Approach) which is an optimized version of BERT which uses the Transformers architecture. However, the method still requires significant improvements in the prediction of types and categories in view of the results obtained 4.3. In addition, the proposed taxonomy covers only 53% of the dataset, so further study is imperative. The study also only covers questions in English. The methodology is new and promising, and it deserves to be considered a research direction, according to the organizers. Some reasons for the poor quality of the results in the prediction of types include:

- Lack of computing power (GPU) for types prediction.

- Unscheduled power outages that prevented the completion of our training.
- Sometimes poor network quality
- -The SPARQL query-based search approach was generally not fruitful, based on our observations.

Given a university-specific knowledge graph, and a set of English questions asked by students, we can apply this model to reduce the number of responses received. This demonstrates the direct impact of these results for society.

4.8 Conclusion

In summary, we were asked to present our experiments. It turns out that our work was based on two major experiments: in the first, we studied the datasets and defined patterns. In the second, we refined the graph. This technique is original according to the organizers, it allowed us to obtain an accuracy of 0.96 for Dbpedia and 0.94 for Wikidata and NDCG of 0.435 However, it is important to specify that this technique has some limitations and requires improvements to outperform other systems.

CHAPTER 5

CONCLUSION

To reply to the research question "For a question in natural language, what is the type of it answer?", we proposed a generic model based on Knowledge Graph refinement techniques. It considers that the prediction of the answer category and type can be reduced to the prediction of links in a knowledge graph. This method was applied on DBpedia and Wikidata dataset provided by SMART organizers. To evaluate our solution, we defined the ground truth as the mean accuracy, category match, NDCG@5, NDCG@10 and mean reciprocal rank (mrr) of the pass challenges. To predict the answer category, we used two scoring functions - one built using statistical learning and the second one built using Naive Bayes. The evaluation on the test data provided by the organizers gave the max accuracy of **0.96%** for DBpedia, category match of **0.94%** for Wikidata. Concerning the prediction of the answer type, we combined Naive Bayes with rule-based graph matching technique. The evaluation on the test data of DBpedia gave NDCG@5 of **0.435** and NDCG@10 of **0.392**. Concerning Wikidata, the mrr was **0.15**. These results allow us to be the first of this challenge over DbPedia (link of ISWC 2022 awards) We note that our NDGC and MRR need significant improvement, and we also propose to tackle other tasks that contribute to the resolution of the question answering problem, which are Entity Linking and Relation Linking.

BIBLIOGRAPHY

[1] iswc2022.semanticweb.org, ""smart challenge 2022"." https://smart-task.github.io/2022/, 2022.

[2] A. Perevalov and A. Both, "Augmentation-based answer type classification of the smart dataset.," in *SMART@ ISWC*, pp. 1–9, 2020.

[3] V. Setty and K. Balog, "Semantic answer type prediction using bert: Iai at the iswc smart task 2020," *arXiv preprint arXiv:2109.06714*, 2021.

[4] A. Jiomekong, V. Tsague, B. Foko, U. Melie, and G. Camara, "Towards an approach based on knowledge graph refinement for answer type prediction," 2020.

[5] A. Perevalov and A. Both, "Mutilingual hierarchical expected answer type classification using the smart 2021 dataset,"

[6] L. Ehrlinger and W. Wöß, "Towards a definition of knowledge graphs.," *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, no. 1-4, p. 2, 2016.

[7] P. Cimiano and H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semant. Web*, vol. 8, p. 489–508, jan 2017.

[8] 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, 2022.

[9] M. Liu, X. Li, J. Li, Y. Liu, B. Zhou, and J. Bao, "A knowledge graph-based data representation approach for iiot-enabled cognitive manufacturing," *Advanced Engineering Informatics*, vol. 51, p. 101515, 2022.

[10] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web*, vol. 8, pp. 489–508, 2017. 3.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[12] S. L. Reed, D. B. Lenat, *et al.*, "Mapping ontologies into cyc," in *AAAI 2002 Conference workshop on ontologies for the semantic Web*, pp. 1–6, 2002.

[13] J. Azanzi, H. Tapamo, and G. Camara, "Combining scrum and model driven architecture for the development of epidemiological surveillance software," 2022.

[14] K. Schwaber, *Agile project management with Scrum*. Microsoft press, 2004.

[15] A. Jiomekong, G. Camara, and M. Tchuente, "Extracting ontological knowledge from java source code using hidden markov models," *Open Computer Science*, vol. 9, no. 1, pp. 181–199, 2019.

[16] A. García-Durán, A. Bordes, and N. Usunier, *Composing relationships with translations*. PhD thesis, CNRS, Heudiasyc, 2015.

[17] R. Busa-Fekete, G. Szarvas, T. Élteto, and B. Kégl, "An apple-to-apple comparison of Learning-to-rank algorithms in terms of Normalized Discounted Cumulative Gain," in *ECAI 2012 - 20th European Conference on Artificial Intelligence : Preference Learning: Problems and Applications in AI Workshop* (D. Raedt, L., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, and P., eds.), vol. 242, (Montpellier, France), Ios Press, Aug. 2012.

[18] S. Bennani, Y. Lakhrissi, G. Khaissidi, A. Mansouri, and Y. Khamlichi, "A new approach for ranking keywords in context of social media," in *WITS 2020: Proceedings of the 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems*, pp. 184–193, Springer, 2020.

APPENDIX

5.1 List of publications in international conferences

1. Jiomekong Azanzi , ***Vadel Tsague*** , Brice Foko , Uriel Melie and Gaoussou Camara . *Towards an Approach based on Knowledge Graph Refinement for Answer Type prediction*

2. Jiomekong Azanzi , Brice Foko , ***Vadel Tsague*** , Uriel Melie and Gaoussou Camara . *Towards an approach based on Knowledge Graph Refinement for Relation Linking and Entity Linking*

3. Azanzi Jiomekong, Cosmas Etoga, Brice Foko, ***Vadel Tsague***, Martins Folefac, Sorel Kana, Mouhamadou Mansour Sow and Gaoussou Camara . *A Large Scale Corpus of Food Composition Tables*