Towards an Approach based on Knowledge Graph Refinement for Relation Linking and Entity Linking

Azanzi Jiomekong[1], Brice Foko[1], Vadel Tsague[1], Uriel Melie[1] and Gaoussou Camara[2]

[1]*Department of Computer Science, University of Yaounde I, Yaounde, Cameroon*

[2]*Unité de Formation et de Recherche en Sciences Appliquées et des TIC, Université Alioune Diop de Bambey, Bambey, Sénégal*

Abstract

This paper presents our contribution to the SMART 3.0 Relation Linking and Entity Linking research problems. This contribution consists of an approach based on knowledge graph refinement techniques for completing the graph with missing entities and relations. The model defined is inspired by the TransE algorithm and a scoring function that defines the distance between two nodes in the graph. The scoring function for Relation Linking is defined using Naive Bayes. Concerning Entity Linking, we identify and extract candidate terms from questions. Thereafter, these terms are used to search for relevant entities in the knowledge graph using LookUp and Wikibase API.

Keywords

Question Answering, Relation Linking, Entity Linking, Knowledge Graphs Refinement, Wikidata, DBpedia

1. Introduction

Knowledge Graph Question Answering (KGQA) is a popular task in the field of Natural Language Processing (NLP) and Information Retrieval (IR). The goal of this task is to answer a natural language question using the facts in the Knowledge Graph (KG) [1]. To build efficient QA systems, several subtasks should be considered. These are: Answer Type prediction, Entity Linking and Relation Linking. While SMART 2020[1] focused on Answer Type (AT) subtask, SMART 2021[2] on Answer Type and Relation Linling (RL) subtasks, SMART 3.0[3] considers Answer Type prediction, Relation Linking and Entity Linking (EL) subtasks. During the SMART 3.0 challenge, we worked on the three subtasks. This paper is presenting our contribution to Entity Linking and Relation Linking subtasks.

Relation Linking consists of predicting relation(s) to use for identifying the correct answer of a question given in natural language. In effect, the ability to know the relations between entities in a user query expressed using natural language allowed us to significantly narrow down the search space for an answer [1]. For instance, given the following question: "Where is University of Yaounde I located?", it is possible to reduce the search space from 40 entities while querying all the 1-hop entities of the "University_of_Yaoundé_I" resource in DBpedia ("dbr:University_of_Yaoundé_I"). This is done by predicting the following relations: *dbo:country*, *dbo:state*, *dbo:city*. Thereafter, these relations are used to translate the question into its corre-

[1]https://smart-task.github.io/2020/
[2]https://smart-task.github.io/2021/
[3]https://smart-task.github.io/2022/

A question with Relation Predicted

```
{
"Question": "Where is University of
    Yaounde I located?",
"relations":[
    "dbo:country",
    "dbo:state",
    "dbo:city"
    ]
}
```

SPARQL Query with RL

```
SELECT DISTINCT * WHERE {
    dbr:University_of_Yaound _I dbo:
        country ?o1 .
    dbr:University_of_Yaound _I dbo:state
         ?o2 .
    dbr:University_of_Yaound _I dbo:city
        ?o3 .
}
```

Result: 3 good results - *:Cameroon*, *:Centre_Region_(Cameroon)*, *:Yaoundé*

SPARQL Query over DBpedia

```
SELECT DISTINCT * WHERE{
    dbr:University_of_Yaound _I
        ?p ?o .
}
```

Result: 40 results

Table 1
SPARQL example on how Relation Linking can help to reduce the solution space over DBpedia dataset

sponding SPARQL query. The illustration is given in table 1. This example shows that Relation Linking is a crucial component to enable QA over Knowledge Graphs.

SMART 2021 was devoted to the Answer Type prediction task and Relation prediction task. However, only 2 (25%) papers were proposed for the relation prediction task. Comparison of related work results are presented by the table 2. This can be normal because relation linking for question answering is known to be a hard task [1]. In effect, some questions have multiple relations, some relations are semantically far and sometimes tokens deciding the relations are spread across the question. On the other hand, some relations are implicit in the text, and there are lexical gaps in relation surface from the KG properties labels.

System	KG	Approach	Techniques	Precision	Recall	F-measure
Khaoula et al. Nadine [2]	Wikidata	BERT	fastai+data augmentation	0.75094	0.8163	0.76018
Khaoula and Nadine [2]	DBpedia	BERT	fastai+data augmentation	0.86475	0.9204	0.86232
Thang et al. [3]	DBpedia	BERT	data oversampling	0.8368	0.8295	0.8315
Thang et al. [3]	Wikidata	BERT	data oversampling	0.6163	0.6110	0.6070

Table 2
Relation Linking results during SMART 2021

On the other hand, Entity Linking [4] consists of identifying entities in a question given

A question with Relation Predicted

```
{
"Question": "Which is the profession of
    Ousmane Sonko?",
"entities":["http://dbpedia.org/resource/
    Ousmane_Sonko"]
}
```

SPARQL Query with EL

```
SELECT DISTINCT * WHERE {
    dbr:Ousmane_Sonko dbp:occupation ?o1 .
}
```

Result: 1 good result - *Politician, Tax inspector*

SPARQL Query over DBpedia

```
SELECT DISTINCT * WHERE{
    ?s dbr:occupation ?o .
}
```

Result: more than 10,000 results

Table 3
SPARQL example on how EL prediction can help to reduce the solution space over DBpedia dataset

in natural language and linking them to the KG, so that they can be used for retrieving the correct answer. In effect, the ability to know the entities hidden in the question of a user query expressed in natural language allows us to significantly narrow down the search space for an answer [4]. For instance, given the following question: "Which is the profession of Ousmane Sonko?" It is possible to reduce the search space from more than 10,000 entities while querying all the 1-hop entities of the "Ousmane_Sonko" resource in DBpedia "dbr:Ousmane_Sonko dbp:occupation" to one good result. This is done by predicting the following entity: http://dbpedia.org/resource/Ousmane_Sonko. Thereafter, this entity is used to translate the user question into its corresponding SPARQL query. The illustration is given in the table 3. This example shows that EL is a crucial component in KGQA systems.

To solve the EL and RL research problems, SMART 3.0 provides us with two versions of large-scale dataset: one dataset is a DBpedia dataset (composed of 760 classes) and the second one is a Wikidata dataset (composed of 50K classes). In addition to these datasets, a restricted vocabulary of all the relations that are used in the datasets for the RL task is provided. Using these datasets we are proposing a solution to the EL and RL problems. This solution is based on Knowledge Graphs refinement techniques [5]. Whatever the method used to construct KGs, it is known that they will never be perfect. Thus, to increase their utility, various refinement methods are proposed to add missing knowledge such as missing entities, missing types of entities, and/or missing relations that exist between entities [5]. To complete a knowledge graph, internal methods use information hidden in the graph and external methods use external knowledge sources such as text corpora, existing vocabularies, ontologies and/or knowledge graphs. In this research, we are using external KG refinement techniques for link prediction and entity discovery. Thus, to solve the RL task, we are considering the restricted vocabulary provided as our external knowledge. To solve the EL task, we consider the DBpedia knowledge graph as our external knowledge and our goal will be to identify terms in the question and map these terms in the graph in order to determine the most relevant entities.

To implement our solutions, we needed to set up a development environment. Thus, we used:

- Operating system: Ubuntu,
- Programming languages: JavaScript,
- JavaScript library: Natural[4].

The source code is provided on GitHub[5].

In the rest of the paper, we present the Relation Linking task in Section 2, the Entity Linking task in Section 3 and the conclusion in Section 4.

2. Relation Linking

To solve the Relation Linking task, we processed as follow: analysis of the dataset (Section 2.1), model definition (Section 2.2) and model training and use (Section 2.3).

2.1. Analysis of the dataset

The aim of this step was to gather, portray and provide a deeper understanding of the structure of the dataset so as to provide efficient solutions. As presented by the organizers, SMART 3.0 datasets are designed to provide one or many relations given a question in natural language.

To understand the RL task 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 main findings from these datasets:

- Data are not equally distributed amongst the different relations;
- Questions types can be divided into *Wh-*, *How many*, *Off which*, *Thus*, *Did*, etc. The taxonomy presented by the Fig. 1 presents the details;
- Questions types can be used to identify a type of relation in the vocabulary.

Given the finding of the analysis of the dataset, we build the question taxonomy, presented by the Fig. 1. This taxonomy will be used to build the model.

2.2. Model definition

The aim of Model definition step was to define a model that can be applied to any dataset to solve the Relation Linking research problem. The first thing we did was to define a graph structure that can be used to represent questions and the relations to predict (see Fig. 2). This consists of matching the taxonomy of the Fig. 1 to the corresponding relations. Thereafter, each question is matched to the corresponding node in the questions taxonomy. The idea is to say that, if a question is linked to a node in the taxonomy and this node is linked to a relation in the vocabulary, thus, this question is linked to this relation. Thus, we defined the *"hasEmbedding"* relation between a question and its embedding in the taxonomy. To define

[4]https://github.com/NaturalNode/natural

[5]https://github.com/jiofidelus/tsotsa/tree/SMART_22

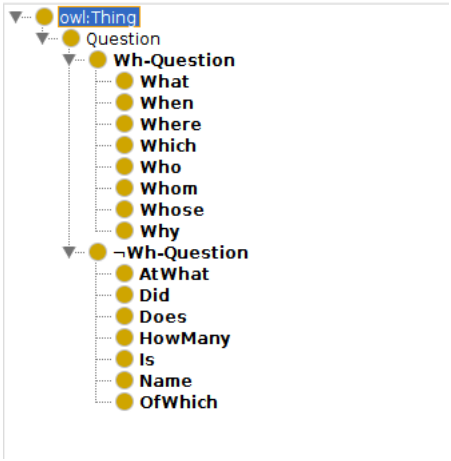

Figure 1: Taxonomy of types of questions build during the analysis of the dataset

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 [6]. The current case consists of extracting terms from unstructured sources which are questions.

We define the Relation Linking dataset as a multi-relation data using a Knowledge graph (see Fig. 2) given by the quadruple $G = (V, E, R, T)$:

- $V = \{v_i \in V\}$ the set of nodes. These nodes are questions, embedding vectors and relations that are hidden in questions;
- $E = \{(v_i, r, v_j)\}$ the set of edges. These 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 the set of relations hidden in this question;
- $T = \{t(v_i)\}$ denotes the node types. The type of nodes relations 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.

In addition to this graph, we suppose that we have an external knowledge source (vocabulary provided by the organizers) that can be used to complete the graph with missing knowledge.

The nodes in the graph of the Fig. 2 contains the following relations:

- ***hasEmbedding:*** this relation defines how closeness a question is with an embedding vector defined in the taxonomy of question. This is very useful because when two questions have the same embedding in the taxonomy, they will have the same relation(s).
- ***"hasRelation":*** represents the relation between a node and a term in the reference vocabulary.

Given this new configuration of the training dataset, we reduced the problem of Relation Linking 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 from the external vocabulary furnished

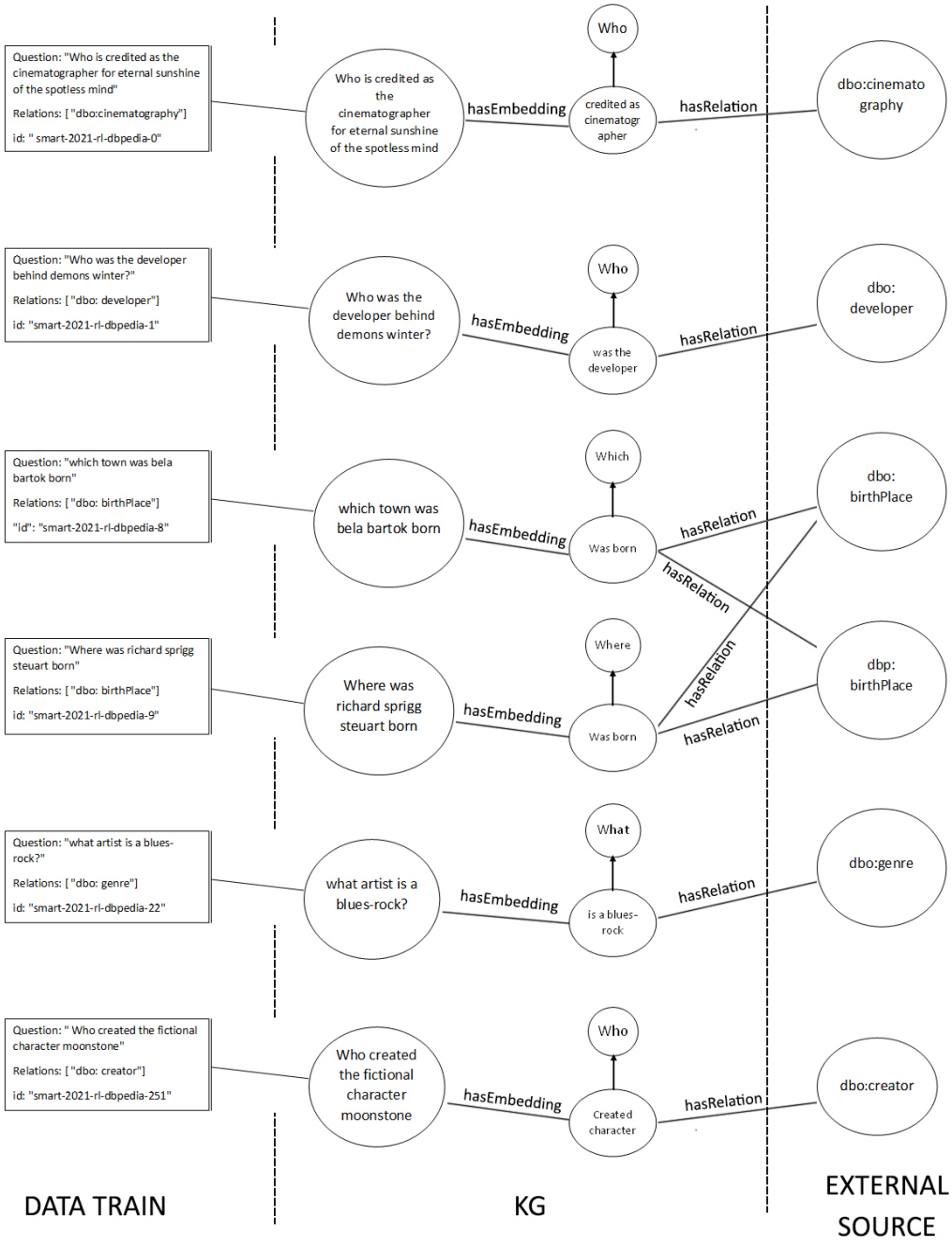

Figure 2: Knowledge graph-based representation of the dataset and the link with external knowledge

by the SMART organizers. Thus, our task is to predict the link between a question and a set of

relations.

Inspired by the TransE algorithm [7] for learning low-dimensional embedding of entities, we defined the model presented by the equation 1. We consider that relationships are represented as translations. The model is defined as a function which takes the node and a relation and predicts all the nodes that are related to this relation. This prediction is based on the proximity between two nodes.

$$\begin{aligned} f_r :& V \times E \times V \longrightarrow \mathbb{R} \\ & f_r(v_i, r, v_j) = ||v_i + v_r - v_j|| \\ & \mathrm{v}_i + v_r \approx v_j \text{ if } f_r(v_i, r, v_j) \approx \beta \end{aligned} \tag{1}$$

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 $f_r(v_i, r, v_j) \approx \beta$ - 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 model that we just defined is based on node embedding, the nodes being the questions and the list of relations of this question. 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 is done using the "question vector (question2vect)" function.

Before the models were trained, the first thing we did was to encode 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 in its title and we define the $question2vect$ function (see equation 2). This function takes a question and returns a vector containing the terms used as the embedding of this question.

$$question2vect(Q) = (t_1, t_2, ..., t_n) \tag{2}$$

In equation 2, $(t_i, i <= n)$ denotes the different terms (made up of words or group of words) in the question title that can be representative of the question. The "question2vect" algorithm is given by the listing 1.

Listing 1: Question2vect algorithm to transform a question into a vector

```
Input:
    Q: Array of String //The words contained in the question
    RL_stopWord: Array of String //The stop words list
Output:
    V: Vector of terms // The Embedded vector
Steps:
    1) Remove all special character inside Q : \,/"#:%
    2) Extract from Q the words in RL_stopWord and put them inside Extr such
        that |size(Q) + size(Extr) - size(RL_stopWord)|< beta // HasEmbedding
    3) Put the results of step (2) in lowercase and delete duplicates
    4) Return the vector V from step (3) with the extracted terms (t_1, t_2,
        ..., t_n)
```

2.3. Models training and use

We used the equation 1 to define specific models for each relations.

- **Learning the "hasEmbedding" relation:** the function $f_{hasEmbedding}$ is given by the equation 3.

$$f_{hasEmbedding}(Q, hasEmbedding, V) = |Q - V| = \alpha \tag{3}$$

To evaluate the function $f_{hasEmbedding}$, we are using the Levenshtein distance. The overall algorithm is given by the listing 2.

Listing 2: f_hasEmbedding algorithm

```
Input:
    Q: Array of String //All words contained in the question title
    V: Vector of terms //The terms contained in the Embedded vector
Output:
    alpha: Int //This is the distance between Q (the question) and V (
        the embedding vector)
Steps:
    1) n = number of string contained in Q
    2) m = number of terms contained in V
    //Recursive implementation of the Levenshtein distance
    3) Calculate the distance between Q and V
        3.1) if m=0 return n
        3.2) if n=0 return m
        3.3) if n=m do HasEmbedded(Queue(Q), Queue(V))
        3.4) else return 1+ min(HasEmbedded(Queue(Q), V), HasEmbedded(Q,
             Queue(V), HasEmbedded(Queue(Q), Queue(V)))
//The function min(x_1, ..., x_n) of step (3.4) returns the minimum
    value of its parameters
//The function Queue(S) takes in parameter the question title and return
     an array of string containing the words whose composed the question
     except the first word
```

To extract knowledge from questions, we used Term Frequency–Inverse Document Frequency (TF-IDF) technique. We identified and removed the less frequent and stop words in questions to obtain the information that can be used to well represent the question. These information were used to define the $f_{hasEmbedding}$ model.
Globally, two vectors are closed if $\alpha \approx \beta$, β being the model parameter defined during the training process. During the experiment, we vary β to obtain different performances.

- ***Learning "hasRelation":*** To predict "hasRelation", relation, we consider that each embedding node has a probability to belong to a set of relations. The scoring function defining this probability can thus be obtained using statistical learning, Naive Bayes, etc. In this research we used Naive Bayes. The algorithm 3 describes how we determine the relations between a node embedding and a set of relations.

Listing 3: f_hasRelation algorithm

```
Input:
    V: a vector of embedded vectors
```

```
    Z: Embedding Matrix //Matrix containing the probability to have a
        Vector v_i (an element of V) link to a Relation R_j
Output:
    R: String //The relation predicted
Steps:
    for each vectors v_i in V
        Z_i = row i from the matrix Z
        for k=0, k < sizeOf(Z_i) //the size of the row
            if (max(|Z_ik|) //Searching of the max probability in the
                row
                return R_k  //Returning the relation having the max
                    probability
```

To train the $f_{hasRelation}$ model, we used DBpedia and Wikidata datasets, with the goal to determine the triples (V, hasRelation, R). This consists of filling the matrix presented by the Fig. 3 with values, so that each value represents the probability that a term in the embedding vector has as relation an element of the vocabulary.

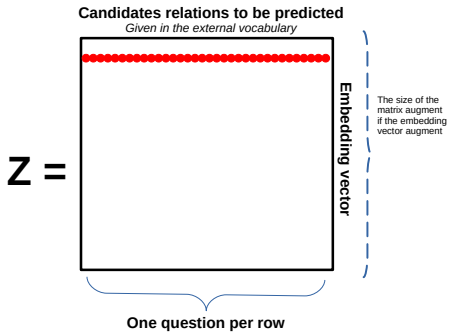

Figure 3: Embedding Matrix

Model refinement consists of extending the taxonomy of question to obtain a larger number of nodes. This is done by varying the parameter β given in the equation 1. To finalize this work, we applied our approach (with embedding vectors of different sizes) on the test data and we got our results analyzed by the SMART 3.0 organizers. The results by considering different embedding vectors are presented by the table 4 for DBpedia and table 5 for Wikidata. In these tables:

Method	**Precision**	**Recall**	**F-Measure**
Naive Bayes	0.37266	0.27052	0.29319
Naive Bayes + WH-taxonomy ($\beta = 1$)	0.37757	0.27602	0.29864
Naive Bayes + taxonomy ($\beta > 4$)	0.38377	0.28017	0.30302
Naive Bayes + taxonomy ($\beta \in [1-4]$)	**0.53426**	**0.47036**	**0.48661**

Table 4
Results of the approach applied to DBpedia dataset

Method	Precision	Recall	F-Measure
Naive Bayes	0.27901	0.31036	0.28541

Table 5
Results of the approach applied to Wikidata dataset

- **Naive Bayes**: we used the Naive Bayes to learn the model without any consideration of the model parameter β;
- **Naive Bayes + WH-taxonomy**: we used the question taxonomy of Fig. 1. In this case, the model parameter β is 1, representing for instance the term "Who" in the question "Who are the gymnasts coached by Amanda Reddin?";
- **Naive Bayes + taxonomy ($\beta > 4$)**: represents the question taxonomy of Fig. 1 extended with more words extracted from the question title. In this case, we took $\beta > 4$ words.
- **Naive Bayes + taxonomy ($\beta \in [1 - 4]$)**: represents the use of the question taxonomy with a value of β fixed between 1 and 4 words.

3. Entity Linking

To solve the EL task, a simple software engineering technique consisting of extracting terms from questions and using these terms to search for entities in the KG were used (see Fig. 4).

To well understand the EL task, we selected ten questions from the training dataset, we identified the entities in the question titles and we compared them to the provided answers. Thereafter, we make simple queries on the DBpedia online using Wikibase[6] and DBpedia LookUp[7] API (the listing 4 presents an example) to search for the same entity in the DBpedia KG.

Listing 4: Example of the use of LookUp and Wikibase API

```
//Querying using LookUp API
https://lookup.dbpedia.org/api/search?query=Ousmane%20Sonko

//Querying using Wikibase API
https://en.wikipedia.org/w/api.php?action=query&generator=allpages&prop=
    pageprops|info&inprop=url&ppprop=wikibase_item&gaplimit=5&gapfrom=Ousmane
    %20Sonko
```

Globally, we reduced the EL task into two sub-problems: entity retrieval in question and entity mapping to the KG (illustrated by the Fig. 4). Thus, for each question Q:

- Search all terms candidates that might be entities in the question Q. In our case, a term can be a word or a group of words. Examples of terms can be "Tchad", "Success Masra".
- Use the terms identified to search for candidates entities in the KG.

If we take the example of the question "Where was Maurice Kamto borned?". From this question, the terms candidates are "Maurice", "Kamto", "was Maurice", "Maurice Kamto", "Kamto

[6] https://www.mediawiki.org/wiki/API:Main_page
[7] https://lookup.dbpedia.org/

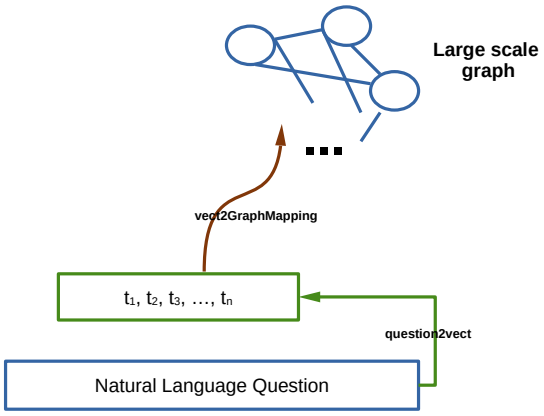

Figure 4: Mapping a question to the knowledge graph

borned", "was Maurice Kamto", "was Maurice Kamto born". From these terms, we realized that the verbs are not pertinent words to find entities and we put in place a filter process to retrieve and remove all the verbs. Applied to our example, we obtain the following candidates: {"Maurice", "Kamto", "Maurice Kamto"}. These three terms are used to query the KG online to search for entities. Once the entities are retrieved, we count the occurrence of each result (see Table 6). The entity having the greatest number of occurrences is selected as the entity we are searching for. In the case of our example, the entity which has 3 occurrences over 1 for the others is Maurice_Kamto.

Entities retrieved	NB occurrence
dbr:Maurice_(emperor)	1
dbr:Mauritius	1
dbr:Joseph_Fotso	1
Maurice_Kamto	3

Table 6
Results of {"Maurice", "Kamto", "Maurice Kamto"} mapping to DBpedia KG

In the example we just presented, the entities were very easy to retrieve. However, the analysis of the dataset showed that the situation is not always so easy. The analysis of the different types of question showed that the terms to seek for entities can be:

1. A proper name: "Success Masra", "Cameroon", "Ousmane Sonko";
2. A common name: "Marketing",
3. A group of words between two stop words.

The cases one and two can be easily solved. However, the third case is the most difficult because one has to determine which are the stop words between the entities. To identify the entities defined by the third case, we made two tables, one composed of the stop words and

the second one composed of stop words that are generally used to link terms and form entities. Thus, for each question, we match the question with the stop word and search for the entity in the KG.

The stop words are used to delimit the names of entities to be searched in the question. The stop word table is created using the set of English language stop words. We have added the term "END_EL" at the end of all the questions. This term is considered as a stop word and is used to complete the stop words table. The overall stop words are used to build a set of term candidates. This approach allowed us to define the different ways an entity can be presented in a question. If we take the example: "where is the University of Yaounde 1 ?" The stop word "of" and "END_EL" are used to define the term "Yaoundé I" and the stop word "the" and "of" are used to define the term "University" as a candidate term to search for an entity in the KG. Given that the word "of" is contained in the second table generally used to link words and that "of" is between two term candidates that are "University" and "Yaounde I", a new term "University of Yaounde I" is added to the list of term candidates. This approach is illustrated by the Fig. 5 and listing 5 presents the algorithm.

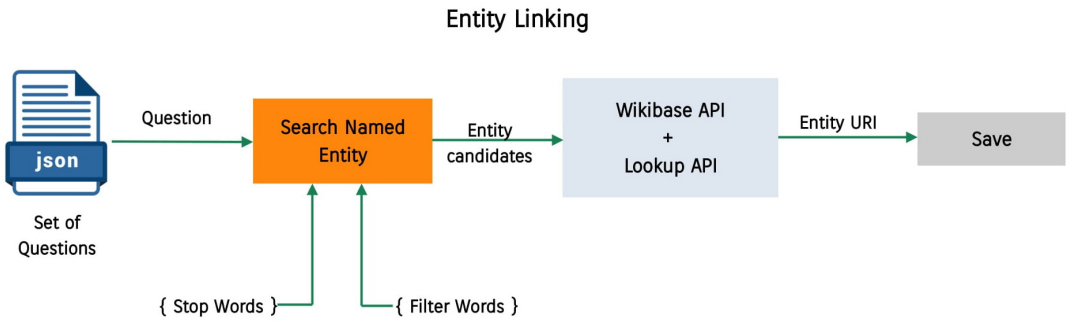

Figure 5: Description of the Entity Linking approach

Listing 5: The algorithm executed to solve the EL task

```
Input:
    Q: Array of question //All the questions from test data
    stopwords: Array of String //The English stop words list. Examples: of,
        the, in, or, etc.
    filterwords: Array of String //The filter words list. Examples: ne, of,
        the
Steps:
    For each question in Q:
        1) Remove the special characters: \,/"#:%
        2) Put all the strings in lower case
        3) Tokenization of result of the step 2
        4) Match tokens obtained in step (3) with {stopwords} to get Entities
            candidates
        //The step 5 allow to obtain the embedding vector V associated to the
            question
```

```
5) Match entities candidates with {filterwords} to produce and clean
    candidates list
7) Use wikibase API and LookUp API to search entities
8) save the entity retrieved
```

We applied this approach on the test data. Table 7 presents the results obtained.

Precision	Recall	F1-score
0.41999	0.54481	0.45729

Table 7
Results of EL task on the test data

We approach the problem of entity linking as software developers. However, this has the following limits: Set-up the stop_words arrays for the question is a difficult task. On the other hand, the number of combinations to determine the entity is very high, making the time complexity very high.

4. Conclusion

To solve the RL and EL tasks, this paper proposes an approach based on KG refinement techniques. This consists of using external knowledge to complete the graph with missing entities and missing relations.

Concerning Relation Linking, we used the vocabulary provided by SMART 3.0 organizers as our external knowledge. Thereafter, we used Naive Bayes to build the embedding matrix that will be used to predict links between a question and its relation(s). The evaluation on the test data gave the max Precision of 0.53426, Recall of 0.47036 and F-measure of 0.48661 for DBpedia and Precision of 0.27901, Recall of 0.31036 and F1-score of 0.28541 for Wikidata.

Concerning Entity Linking, we used the KG itself as external knowledge. Thus, our goal was to identify the most relevant terms (one word or a group of words) that can be matched to the KG and from these terms, determine relevant entities. The EL task was used on DBpedia dataset only and the evaluation by the SMART 3.0 organizers gave the Precision of 0.41999, the Recall of 0.54481 and the F1-score of 0.45729.

We started this challenge with no prior knowledge on Entity Linking and Relation Linking tasks and we proposed a solution. Future work consists of exploring different machine learning, deep learning and natural language processing models that can be used to help us improve the quality of our results. We particularly found good papers [8, 9] on relation linking and entity linking [9, 10, 11]. We are planning to explore and test these methods to solve the Entity Linking and Relation Linking tasks.

Acknowledgments

We would like to thank the ISWC conference and the SMART challenge organizers and all the reviewers. This work was partially supported by Neuralearn.ai (Start-up Cameroon).

References

[1] G. Rossiello, N. Mihindukulasooriya, I. Abdelaziz, M. Bornea, A. Gliozzo, T. Naseem, P. Kapanipathi, Generative relation linking for question answering over knowledge bases, 2021. URL: https://arxiv.org/abs/2108.07337. doi:10.48550/ARXIV.2108.07337.

[2] Reaching out for the Answer: Relation Prediction, volume 3119 of *SeMantic Answer Type and Relation Prediction Task at ISWC 2021 Semantic Web Challenge (SMART2021)*, CEUR Workshop Proceedings, 2021.

[3] The Combination of BERT and Data Oversampling for Relation Set Prediction, volume 3119 of *SeMantic Answer Type and Relation Prediction Task at ISWC 2021 Semantic Web Challenge (SMART2021)*, CEUR Workshop Proceedings, 2021.

[4] W. Zhang, W. Hua, K. Stratos, Entqa: Entity linking as question answering, CoRR abs/2110.02369 (2021). URL: https://arxiv.org/abs/2110.02369.

[5] P. Cimiano, H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, Semant. Web 8 (2017) 489–508. URL: https://doi.org/10.3233/SW-160218. doi:10.3233/SW-160218.

[6] A. Jiomekong, G. Camara, M. Tchuente, Extracting ontological knowledge from java source code using hidden markov models, Open Computer Science 9 (2019) 181–199.

[7] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, p. 2787–2795.

[8] G. Rossiello, N. Mihindukulasooriya, I. Abdelaziz, M. Bornea, A. Gliozzo, T. Naseem, P. Kapanipathi, Generative relation linking for question answering over knowledge bases, in: The Semantic Web – ISWC 2021: 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24–28, 2021, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2021, p. 321–337. URL: https://doi.org/10.1007/978-3-030-88361-4_19. doi:10.1007/978-3-030-88361-4_19.

[9] J. a. Gomes, R. C. denbsp;Mello, V. Ströele, J. F. denbsp;Souza, A study of approaches to answering complex questions over knowledge bases, Knowl. Inf. Syst. 64 (2022) 2849–2881. URL: https://doi.org/10.1007/s10115-022-01737-x. doi:10.1007/s10115-022-01737-x.

[10] C. Ran, W. Shen, J. Gao, Y. Li, J. Wang, Y. Jia, Learning entity linking features for emerging entities, 2022. doi:10.48550/ARXIV.2208.03877.

[11] W. Zhang, W. Hua, K. Stratos, Entqa: Entity linking as question answering, 2021. URL: https://arxiv.org/abs/2110.02369. doi:10.48550/ARXIV.2110.02369.