Indonesian Journal of Electrical Engineering and Computer Science
Vol. 36, No. 2, November 2024, pp. 983~993
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v36.i2.pp983-993

Applying inductive logic programming to automate the function of an intelligent natural language interfaces for databases

Hanane Bais[1], Mustapha Machkour[2]
[1]Laboratory of LAMIGEP, EMSI Marrakech, Marrakech, Morocco
[2]Department of Computer Sciences, Ibn Zohr University, Agadir, Morocco

Article Info

Article history:

Received Dec 13, 2023
Revised Jul 6, 2024
Accepted Jul 14, 2024

Keywords:

Artificial intelligence
Database
Inductive logic programming
Linguistic operations
Natural language processing

ABSTRACT

One of the foundational subjects in both artificial intelligence (AI) and database technologies is natural language interfaces for databases (NLIDB). The primary goal of NLIDB is to enable users to interact with databases using natural languages such as English, Arabic, and French. While many existing NLIDBs rely on linguistic operations to meet the challenges of user's ambiguity existing in natural language queries (NLQ), there is currently a growing emphasis on utilizing inductive logic programming (ILP) to develop natural language processing (NLP) applications. This is because ILP reduces the requirement for linguistic expertise in building NLP systems. This paper outlines a methodology for automating the construction of NLIDB. This method utilizes ILP to derive transfer rules that directly translate NLQ into a clear and unambiguous logical query, which subsequently translatable into database query languages (DQL). To acquire these rules, our system was trained within a corpus consisting of parallel examples of NLQs and their logical interpretations. The experimental results demonstrate the promise of this approach, as it enables the direct translation of all NLQs with grammatical structures similar to those already present in the trained corpus into a logical query.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Hanane Bais
Laboratory of LAMIGEP, EMSI Marrakech
Marrakech, Morocco
Email: H.bais@emsi.ma

1. INTRODUCTION

Intelligent natural language interfaces for databases (NLIDB) [1] is an active research area in the field of natural language processing (NLP) and artificial intelligence (AI) [1], [2]. NLIDB's main objective is to improve human-database interaction by supporting users in information extraction from databases by writing simple queries in natural language [3], [4] without requiring expertise in database query languages (DQL) [5]. In order to convert natural language queries (NLQ) into a clear logical interpretation, the majority of NLIDBs have up to now depended on linguistic expertise [6], [7]. But using linguistic analyses to create NLP systems is still a very difficult and involved process and, NLIDB systems based on linguistic approaches utilize techniques of syntactic and semantic analysis to understand the structure and meaning of NLQ. Additionally, these systems often rely on predefined grammatical rules and semantic models to translate NLQ into database queries. These systems often require domain-specific or language-specific rules, it may make their maintenance expensive and difficult. One efficient method to automate the process of development of NLP systems is the using of the inductive logic programming (ILP) [8]. NLIDB systems based on programmation logique inductive (PLI) employ machine learning techniques to induce rules from examples of NLQ and their corresponding DQL equivalents. Furthermore, these PLI-based systems aim to automate the construction of the natural language interface by learning rules directly from data, thus reducing reliance on manual linguistic rules. Additionally, PLI-based systems may exhibit greater adaptability to various domains or languages since they learn from data rather than predefined rules. Moreover, by learning from data, PLI systems can reduce dependency on prior linguistic analyses, potentially enhancing the robustness and accuracy of translations. For that, we propose in this paper anew method based on supervised machine learning to automate the function of NLIDB. This method consiststo learn rules that map directly the NLQ into a logical interpretation. To induce such rules, the proposed system trained within a corpus composed of parallel examples of NLQs and their logical interpretations. Normally to generating the transfer rules, we need enormous human effort. But, the richness of the predicate logic (first order logic) used in ILP [8], [9] we offer helpfully provide advantages during the developing of our interface.

One of the greatest advantages of the proposed interface is its ability to function independently of the database domain and model [10]. And, it may offer greater adaptability, reduced linguistic dependency, and automation of interface construction. However, they often require labeled datasets for learning and may necessitate expertisein machine learning for implementation.

The rest of the paper is organized as follows. Some previous systems are cited first. An overview of ILP is presented. Afterward, the architecture of the proposed system is presented. Then, we will present the experimental results of using ILP on the NLIDB. Finally, we conclude and suggest some possible future research directions.

2. RELATED WORK

Research into NLIDB began in the late 1960s and early 1970s [11]. From that time, the majority of NLIDBs have utilized linguistic skills to convert NLQs into a logical representation [12], [13]. Nevertheless, this approach typically requires considerable time, and hand-crafted parsers often struggle with robustness issues (e.g. it generalizes feebly on novel sentences) [14]. That’s why there is an increasing focus on using ILP technique to construct automate NLP systems [15]. The application of the ILP for developing NLP is not a new topic. ILP has been used in many NLP domains like part-of-speech tagging [15], morphological analysis [16] and word segmentation [17]. Concerning the application of ILP in the realization of NLIDBs, there have been some contributions.

The initial contribution of the application of ILP for developing NLIDB is a system called Chill [18]. Chill is an acquisition system for parsers aimed at developing an automated NLIDB. It achieves parser acquisition by acquiring search-control rules using a logic program that embodies a shift-reduce parser. Employing ILP techniques, Chill learns relational control knowledge. Starting with a broad framework to construct suitable logical forms, Chill can be trained on a corpus containing natural language sentences paired with their corresponding database queries. Consequently, it enables parsers to translate NLQs into database queries [19].

The second contribution is proposed by Tang. Tang presented a machine learning approach to semi-automate the realastion of NLIDB [20]. He introduced a 'meta' ILP learning approach, which combines the strengths of various ILP learners to induce a semantic parser. This approche outperforms the use of a single learner. One of the problems with all ILP contributions in ILBD is that they do not demonstrate any domain independence. Furthermore, all the exiciting NLIDBs are designed for a specific database model. This requires reconfiguration with each new domain and model [21]. The problem of applying ILP for NLIDB independence from the database model has never been a research subject. In this paper, we propose leveraging ILP to automate the operation of NLIDB systems. This method involves acquiring rules that directly translate user queries presented in natural language into a clear interpretation expressed in XML format: the XML logical query (XLQ).

The primary advantage of our system over others is that it does not require reconfiguration for a different database. This is achievable through the use of a machine learning-based method, enabling it to enhance its knowledge base during execution and consequently adapt to multiple domains. In general, whenever our system transitions to a new domain, this method provides it with the associated concepts. Furthermore, it’s worth noting that the proposed interface is not specific to a particular database model. This adaptability is attributed partly to the complete separation between the linguistic processing phase of the natural language input and the query generation phase in NLIDB, and partly to the formalization of logical queries into XML, which ensures portability across different models. Finally, our proposed system benefits from his experience in inducing automatically rules that directly map all NLQs with grammatical structures similar to those already present in the trained corpus into a logical query.

3. PROPOSED METHOD

The performance level of a NLIDB varies considerably. The most robust systems operate within a limited domain and model. Transitioning from an NLIDB built for a particular domain or model to another one requires reconfiguration. To reduce the cost of this configuration. Often, one of the solutions used in this research is based on linguistic operations.

Our method consists of proposing a generic NLIDB that use ILP and operates independently of database domain and model. By using the ILP, our system can automatically map the natural language to an unambiguous logical interpretation. Furthermore, the proposed interface is not specific to a particular database domain. This capability is due to the complete separation between the database knowledge module and other modules. Also, the formalization of logical queries in XML, which provides portability to different models. The Figure 1 displays the proposed architecture of our system.

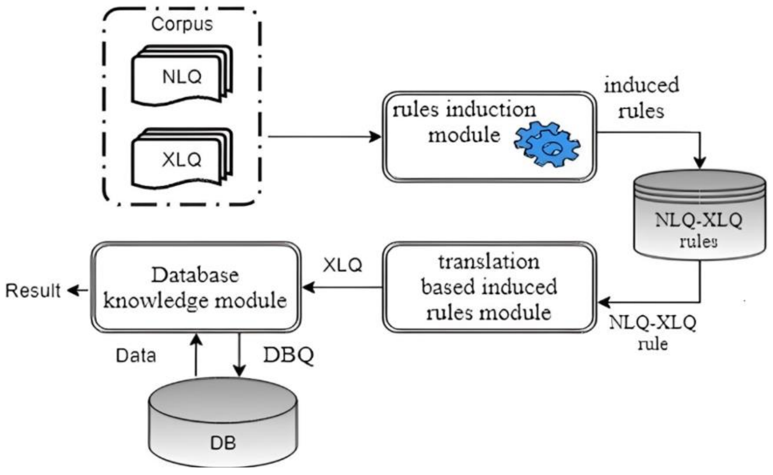


Figure 1. The architecture of system

As shown in Figure 1, we can divide the proposed architecture into three modules:
- The rules induction module: using the supervised machine learning technique to induce NLQ-XLQ rules
- The translation based induced rules module: application of the appropriate NLQ-XLQ rule to create the XLQ
- Database knowledge module: translating the XLQ into DBQ

In the following, we explain in-depth the functioning of each module.

4. METHOD

4.1. Rule induction module

In this module, we address the using of the supervised machine learning technique to induce NLQ-XLQ rules. Before the induction of these rules, our system trained parallel corpus of a set of NLQ paired with XLQ. The rule induction module tacks NLQ paired with the XLQ as input then it applies ILP technique to produce NLQ-XLQ rules as output. We should mention that the negative example of NLQ paired with XLQ will not be accessible and that it’s intractable to use the closed world assumption to explicitly generate negative examples given the large number of possible NLQ and XLQ. Furthermore, it is agreed that to acquire language, children are exposed to modest if any negative feedback [22]. That’s why it’s essential to propose a method to learning without explicit negative examples [23]. Fortunately, numerous ILP methods have been proposed learning only from positive examples; this is possible if the target predicate represents a function (which is ourcase) or if the training. Example is in some sense complete. The Figure 2 displays the function of rule inductionmodule

As shown in Figure 2, the first step in the process of inducing NQL-XLQ rules is the partition of the NLQ (tokenization) [24]. In this phase, our system divides the NLQ into tokens to simplify the NLQ and deal with a token rather than an entire sentence. After the tokenization step, the part of speech tags applied to recognizethe grammatical structure of the NLQ. Figure 3 shows the output of tokenization and part of speech tags of the NLQ:
Give name of all client where age >25

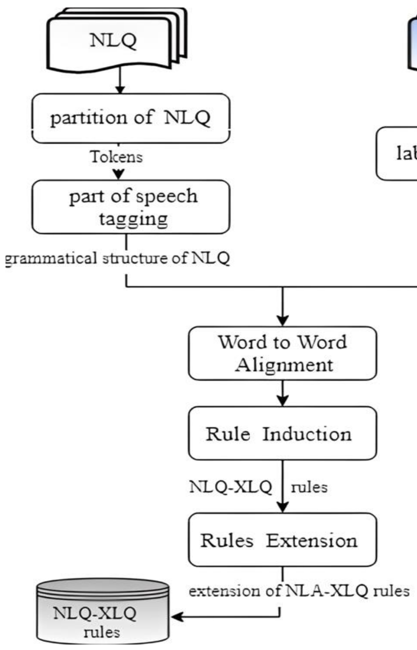


Figure 2. Process of inducing NQL-XLQ rules

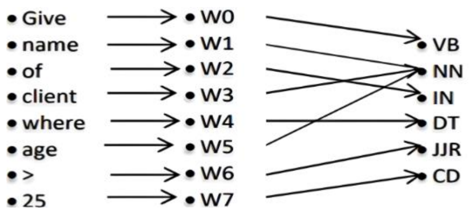
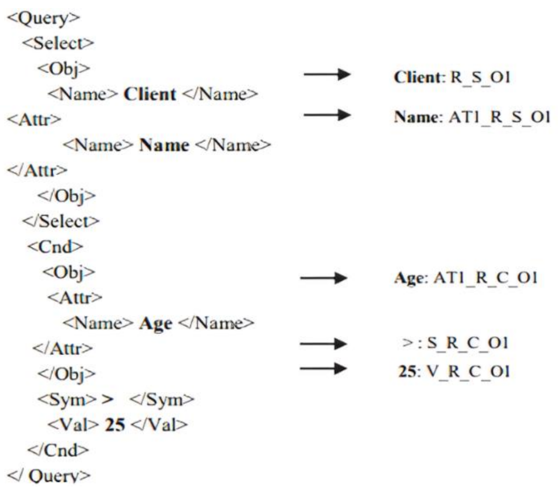


Figure 3. Tokenization of NLQ

We have developed a specific method for labeling the XLQ based on the placement of words within it. Here are some examples of the labels we suggest:

– R_S_O1= the initial object within the select fraction of the query;
– AT1_R_S_O1= the first attribute of the initial object in the select part of the query.

The result of labeling the XLQ corresponding to the NLQ in the Figure 4 is shown in next figure:



```
<Query>
  <Select>
    <Obj>                                    Client: R_S_O1
      <Name> Client </Name>
<Attr>                                       Name: AT1_R_S_O1
        <Name> Name </Name>
</Attr>
    </Obj>
  </Select>
  <Cnd>
    <Obj>                                    Age: AT1_R_C_O1
     <Attr>
        <Name> Age </Name>
   </Attr>                                    > : S_R_C_O1
    </Obj>                                   25: V_R_C_O1
    <Sym> >  </Sym>
     <Val> 25 </Val>
  </Cnd>
</ Query>
```

Figure 4. Labeling the XLQ

A prerequisite step for the induction of NLQ-XLQ rule is the word-to-word alignment [25]. In this phase, the result of the part of speech tags of NLQ and labeling of XLQ is used to identify the equivalent words of the NLQ and the XLQ. In Figure 5, we display the result of word-to-word alignment.

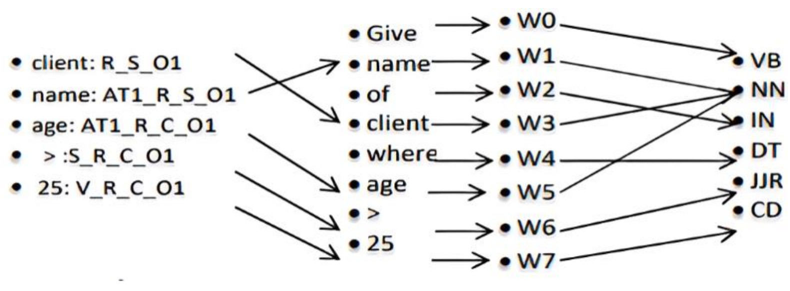


Figure 5. Word-to-word alignment

To induce the NLQ-XLQ rules, the next step is to apply the unification between the NLQ and XLQ sequences. The XLQ is identified through the application of the labeling function, while the NLQ is identified by its grammatical structure. The NLQ-XLQ rule aligns the NLQ 's grammatical structure with the structure of XLQ, as depicted in Figure 6.

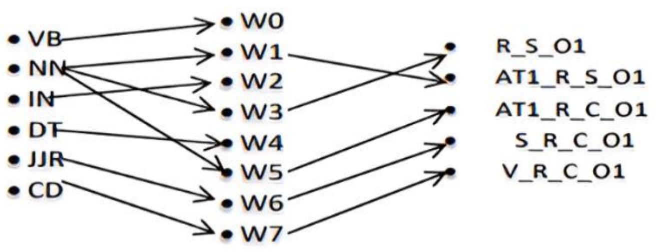


Figure 6. Induction of the NLQ-XLQ

The next equation represents the global of the NLQ-XLQ rule:

$$\sum_{j=0}^{n} \quad (W_j : GC_j) \rightarrow \sum_{k=0}^{l} \quad (W_j : E_k)$$

with:
GCj: is the grammatical category of the token Wj in NLQ;
n: number of tokens in NLQ;
Ek: is the label of the fractions Fj in XLQ;
l: number of fractions in XLQ;
WJ: the order of word in NLQ.
The induced NLQ-XLQ rule from example 1 is the following:

$$W0: VB + W1: NN + W2: IN + W3: NN + W4: DT + W5: NN + W6: JJR + W7: CD$$
$$\rightarrow$$
$$W3: R\ S\ OI + W\ I: AT\ I\ R\ S\ OI + W\ 5: AT\ I\ R\ C\ OI + W\ 6: S\ R\ C\ OI + W\ 7$$

To enable our system to benefit from its experience and translate a large number of NLQs directly into XLQs, we use a technique that helps to create extensions of the NLQ-XLQ rules. With this method, our system can add new rules to the rules to the existing ones without needing to prepare the example that will induce these rules. The extension rules represent instances of the induced rules. The extension of the induced NLQ-XLQ is created using the grammar shown in Figure 7. The Figure 8 displays the extension of the NLQ-XLQ rule induced from example 1:

```
VBW → VB | VBG | VBD | VBN | ε
NNW → NN | NNS | NNP | NNPS
JJW→ JJ | JJR | JJS | ε
WPW → WP | WDT | WPS | WRB | WD |ε
PRW → PRP | PRP$ |ε
RB → RB | RBR| RBS| ε
DTW → DT | ε
```

Figure 7. Grammar for the extension of NLQ-XLQ rule

```
W0: VB+ W1: NNS+ W2: IN+ W3: NNPS + W4: DT+ W5: NN+ W6: JJR
→
W3:R_S_O1+W 1:AT_1_R_S_O1+W 5:AT_1_R_C_O1+W 6: S_R_C_O1
Extension 2:
W0: VBN+ W1: NN+ W2: IN+ W3: NNS + W4: DT+ W5: NN+ W6: JJR
→
W3:R_S_O1+W 1:AT_1_R_S_O1+W 5:AT_1_R_C_O1+W 6: S_R_C_O1
Extension 3:
W0: VBG+ W1: NN+ W2: IN+ W3: NNP + W4: DT+ W5: NNS+ W6: JJR
→
W3:R_S_O1+W 1:AT_1_R_S_O1+W 5:AT_1_R_C_O1+W 6: S_R_C_O1
Extension 4:
W0: VBD+ W1: NN+ W2: IN+ W3: NN + W4: DT+ W5: NN+ W6: JJR
→
W3:R_S_O1+W 1:AT_1_R_S_O1+W 5:AT_1_R_C_O1+W 6: S_R_C_O1.
```

Figure 8. Extension of the NLQ-XLQ

4.2. Translation based induced rules module

The module for translation based on induced rules operates through four steps. Initially, it segments the NLQ into tokens. Next, it utilizes the part-of-speech tags of these tokens to determine their grammatical structure. Subsequently, the system searches for an NLQ-XLQ rule that matches the grammatical structure of the NLQ among the previously induced NLQ-XLQ rules. Finally, our system uses the adequate NLQ-XLQ rule for the generating the corresponding XLQ. The Figure 9 shows the translation based induced rules process.

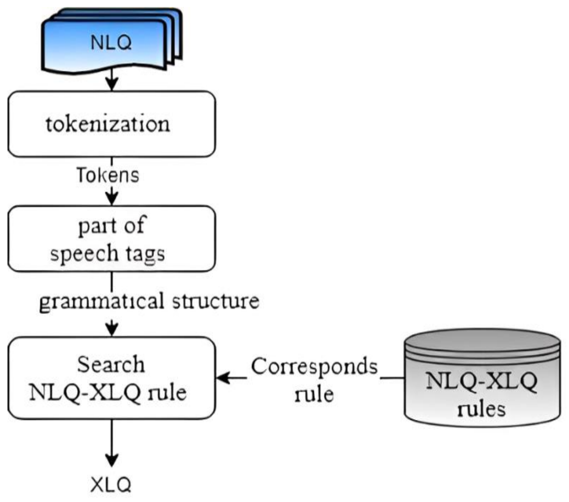


Figure 9. Translation based induced rules

The next example displays how the induced rule can be used to translate The NLQ input into an XLQ output.

Input NLQ: show addresses of students whose marks=14

1) The part of speech tags of input NLQ is displays in Figure 10.

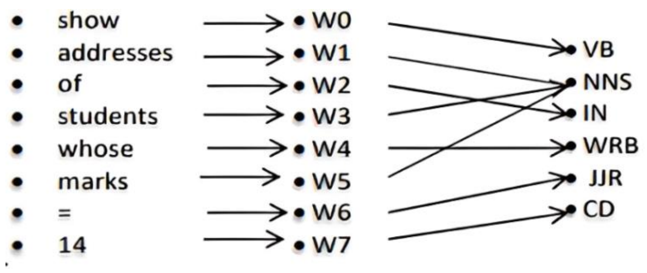


Figure 10. Part of speech tags result of input NLQ

2) Search the NLQ-XLQ rule whose left side matches the grammatical structure as the input NLQ

– Left part of needed NLQ-XLQ rule:

$$W0: VB + W1: NNS + W2: IN + W3: NNS + W5: WRB + \ W6: NN + \ W7$$

– Corresponds NLQ-XLQ rule:

$$W0: VB + \ W1: NN + \ W2: IN + \ W3: NN + \ W4: DT + \ W5: NN + \ W6: JJR + \ W7: CD$$
$$\rightarrow$$
$$W3: R\ S\ OI + W\ I: AT\ I\ R\ S\ OI + W\ 5: AT\ I\ R\ C\ OI + W\ 6: S\ R\ C\ OI + W\ 7$$

3) The generated XLQ is shown in Figure 11.

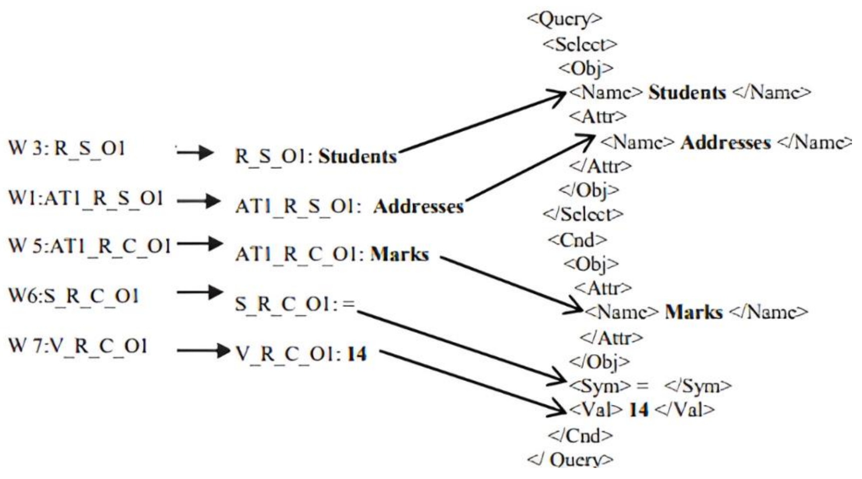


Figure 11. The output XLQ

4.3. Database knowledge module

The objective of this module is to translate XLQ into DBQ by mapping each element of the XLQ to its corresponding clause in the DQL. Our system operates independently of the database model (XML and relational). For relational databases, the system generates an SQL query, while for XML databases, it generates an XPATH query.

The generation of DQL involves three steps, each handling a specific part of the DBQ. First, the system processes the XLQ to extract attribute names for constructing the SELECT clause. Next, it constructs the FROM clause by identifying table names within the XLQ. Finally, the system extracts selection conditions from the XLQ to build the WHERE clause. Concatenating the results of these steps produces the DBQ. During each DQL construction step, our system verifies the validity of table and attribute names retrieved from the XLQ against the database dictionary. If they are invalid, a mapping table containing synonyms of the names of table and attribute is utilized. This feature allows users to write NLQ without requiring precise knowledge of database table and attribute names. After the generation of the DBQ, the system submits it to the database management system (DBMS) and presents the result of SQL queries in tabular form or XML format for XPATH queries. An example of DQL generation is illustrated in Figure 12.

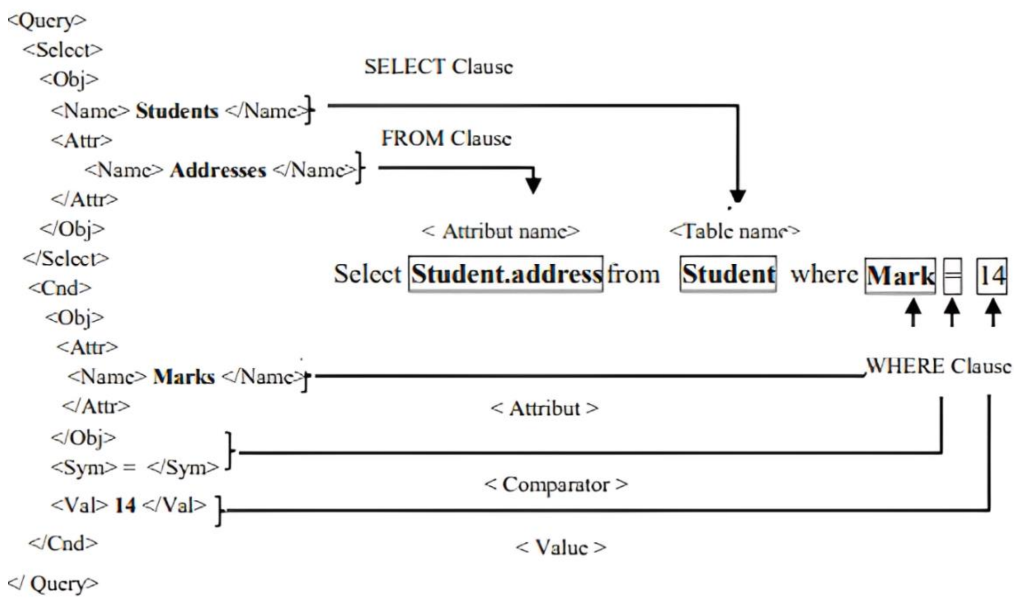
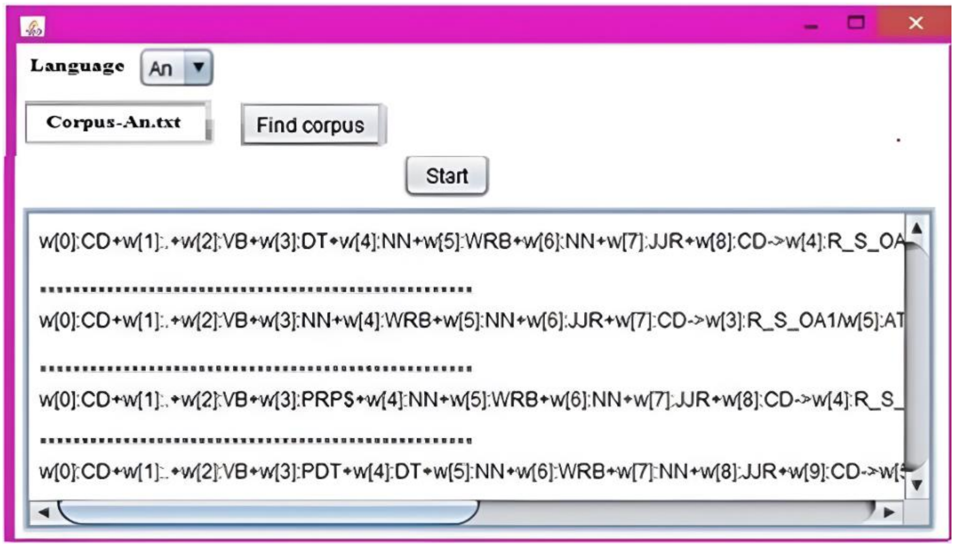


Figure 12. Translating the XLQ into DBQ

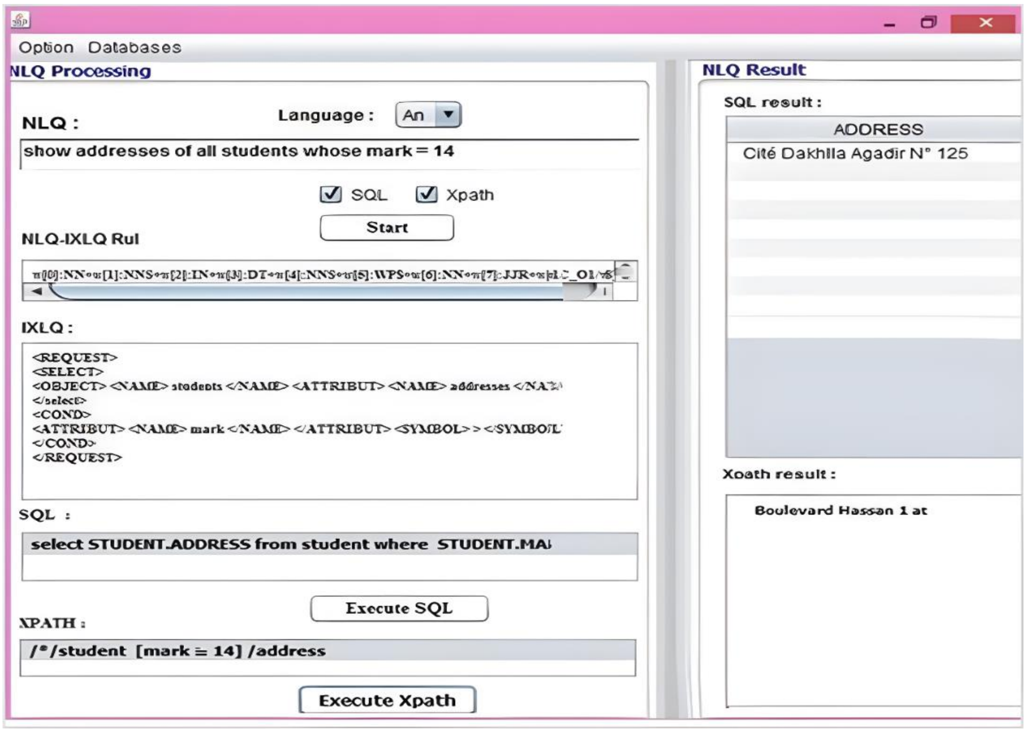
5. RESULTS AND DISCUSSION

The interface in Figure 13 illustrates the process of training the NLQ-XLQ corpus to induce NLQ-XLQ rules. The interface in Figure 14 represents the translation based induced rule. The Table 1 shows the output of translating some NLQ using NLQ-XLQ rules. We have tested with many database domains (database of students, projects, and employees). Additionally, we have tested the functions with two models: relational and XML databases.



```
w[0]:CD+w[1]:.+w[2]:VB+w[3]:DT+w[4]:NN+w[5]:WRB+w[6]:NN+w[7]:JJR+w[8]:CD->w[4]:R_S_OA
.....................................................
w[0]:CD+w[1]:.+w[2]:VB+w[3]:NN+w[4]:WRB+w[5]:NN+w[6]:JJR+w[7]:CD->w[3]:R_S_OA1/w[5]:AT
.....................................................
w[0]:CD+w[1]:.+w[2]:VB+w[3]:PRP$+w[4]:NN+w[5]:WRB+w[6]:NN+w[7]:JJR+w[8]:CD->w[4]:R_S_
.....................................................
w[0]:CD+w[1]:.+w[2]:VB+w[3]:PDT+w[4]:DT+w[5]:NN+w[6]:WRB+w[7]:NN+w[8]:JJR+w[9]:CD->w[
```

Figure 13. Training the NLQ-XLQ corpus



NLQ :
show addresses of all students whose mark = 14

NLQ-IXLQ Rul
π[0]:NN∘π[1]:NNS∘π[2]:IN∘π[3]:DT∘π[4]:NNS∘π[5]:WPS∘π[6]:NN∘π[7]:JJR∘π[8]:C_O1/

IXLQ :

```
<REQUEST>
<SELECT>
<OBJECT> <NAME> students </NAME> <ATTRIBUT> <NAME> addresses </NA
</select>
<COND>
<ATTRIBUT> <NAME> mark </NAME> </ATTRIBUT> <SYMBOL> > </SYMBOL
</COND>
</REQUEST>
```

SQL :
select STUDENT.ADDRESS from student where STUDENT.MA

XPATH :
/*/student [mark = 14] /address

SQL result :

ADDRESS
Cité Dakhla Agadir N° 125

Xpath result :
Boulevard Hassan 1 at

Figure 14. Translation based induced rules

Table 1. NLQs with aggregate function

NLQ	Generated SQL	Generated XPATH
Show customers	SELECT * FROM customer	/*/ customer /*
Students	SELECT * FROM student	/*/ student /*
Give all the salary of our employees	SELECT salary FROM employee	/*/employee/ salary
Ages, emails and faxes of customers	SELECT age, email, faxe FROM customer	customers /*/ customers /email /*/ customers /age /*/ customers /faxedress
Give me the projects names	SELECT email FROM client	/*/client/email
And customers emails	SELECT name FROM project	/*/ project /name
Show me the customers whose name is 'Hanane' or 'Mustapha'	SELECT* FROM customer where name in ('Mustapha', 'Hanane')	/*/ customer [name="hanane"]/* /*/ customer [name="Mustapha"]/*
What is the information of customers whose age is more than 40?	Select * FROM customer where age > 40	/*/ customer [age > 40] /age
Count the number of projects	SELECT COUNT (*) AS NB-project	/*/count(project)
Show me the max age of clients where age is lessthan 40	Select * FROM customer where age > 40	/*/ customer [age > 40] /age
Count the number of projects	SELECT COUNT (*) AS NB-project FROM project	/*/count(project)
Show me the max age of clients where age is lessthan 40	SELECT MAX (age) AS max-age FROM client where age < 40	max(/*/client[age <40]/age)
Display the max mark and the min age of students	SELECT max (mark) AS max--mark, min (age) as min -age FROM student	max(/*/student/mark) |min(/*/student /age)
Show the max and themin age of customers where age > 40	SELECT max (client.age) as max-client.age, min (client.age) as min-client. age FROM client where age > 40	max(/*/client [age>40]/age)min(/*/client[age >40]/age)
Give the name of student with the max mark?	SELECT name FROM student where mark in (SELECT max (mark) FROM student)	/*/student[marks=max (/*/student/ marks)]/nam

We evaluated the performance of our system using a set of 1,000 NLQs. The results from this test are summarized in Table 2. From Table 2, it can be seen that out of 1,000 NLQs evaluated, the system successfully generated an SQL query for 955 of them. 95.2% of these NLQs were converted into XPATH searches. Furthermore, 92.46% of the SQL translations were accurate, while 896 NLQs were incorrectly converted into XPATH queries. Additionally, the evaluation shows that 94.56% of the correctly generated SQL queries and 89.95% of the XPATH queries matched their corresponding NLQs. Table 3 illustrates the improvement in response time in second (s)when using NLQ-XLQ rules. As show in Table 3 The response time is reduced significantly, reaching up to 93.77% in certain cases, when using ILP compared to that of linguistic operations. Surely, this enhances the performance of our system.

To demonstrate the consistency in the use of ILP and to avoid memory overload and high CPU usage, we studied how many rules have been generated during the training of 100 NLQs. The result obtained is illustrated in Figure 15. From Figure 15, it can be seen that our system provides logarithmic growth during the rule generation. This degrades newly induced rules after a certain number of training sessions, which demonstrates the consistency of this system.

Table 2. Answered NLQ/DBQ correctly translated/DLQ matches NLQ

	SQL	XPATH
Answered queries	955 (95.5%)	952 (95.2%)
Correctly generated	883 (92.46%)	896 (94.11%)
DLQ matches NLQ	835 (94.56%)	806 (89.95%)

Table 3. Improvement in response time

Type of NLQ	NLQ-XLQ rule	linguistic expertise	%
NLQ without projection and selection	4.18 (s)	0.26 (s)	93.77
NLQ with projection and without selection	5.97 (s)	0.70 (s)	88.27
NLQ without projection and with selection	5.61(s)	1.04 (s)	81.46
NLQ with projection and selection	5.86 (s)	0.91 (s)	84.74
NLQ with aggregate functions	6.08 (s)	1.32 (s)	78.28

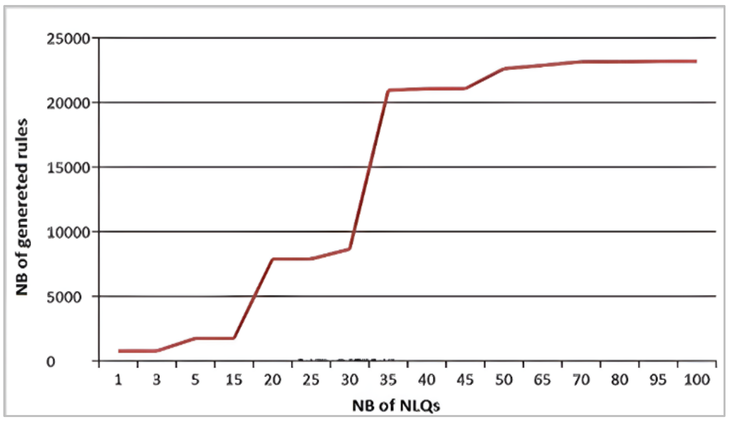


Figure 15. Logarithmic growth during the inducing of rules

6. CONCLUSION

This paper describes the exploitation of a supervised machine learning technique to develop an example based transfer tool that automatically induces transfer rules from the example of NLQ paired with their logical interpretation. We have presented a framework using ILP to learn rules that map NLQ into XLQ by a training corpus of NLQ paired with XLQ. The results of experimentation demonstrate the ability to learn a very important number of rules that produce a correct answer to a user’s NLQ. This approach does not require reconfiguration when switching to a different database domain. Furthermore, it was not designed for a specific database model. This suggests flexibility and adaptability across different database domains and models. This flexibility can be advantageous in scenarios where it is necessary to work with different types of databases without significant system modifications or adjustments. Moreover, our proposed interface can benefit from his experience in the automatic induction of rules that directly map all NLQs with grammatical structures similar to those already present in the corpus. A potential future direction is to further enhance the abilities of our system to understand NLQ in other languages and expand our system's ability to interface with other database models.

REFERENCES

[1] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, “Natural language interfaces to databases – an introduction,” *Natural Language Engineering*, vol. 1, no. 1, pp. 29–81, Mar. 1995, doi: 10.1017/s135132490000005x.
[2] A. Colas, T. Bui, F. Dernoncourt, M. Sinha, and D. S. Kim, “Efficient deployment of conversational natural language interfaces over databases,” 2020, doi: 10.18653/v1/2020.nli-1.4.

[3] R. Akerkar and M. R. Joshi, "Natural language interface using shallow parsing," *International Journal of Computer Science and Applications*, vol. 5, no. 3, pp. 70–90, 2008.

[4] M. Llopis and A. Ferrández, "How to make a natural language interface to query databases accessible to everyone: An example," *Computer Standards &amp; Interfaces*, vol. 35, no. 5, pp. 470–481, Sep. 2013, doi: 10.1016/j.csi.2012.09.005.

[5] A.-M. Popescu, O. Etzioni, and H. Kautz, "Towards a theory of natural language interfaces to databases," 2003, doi: 10.1145/604050.604070.

[6] R. J. Mooney, "Inductive logic programming for natural language processing," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 1997, pp. 1–22.

[7] G. Rao, C. Agarwal, S. Chaudhry, and N. Kulkarni, "Natural language query processing using semantic grammar," vol. 2, pp. 219–223, 2010.

[8] N. Lavrač, S. Džeroski, and M. Numao, "Inductive logic programming for relational knowledge discovery," *New Generation Computing*, vol. 17, no. 1, pp. 3–23, Mar. 1999, doi: 10.1007/bf03037580.

[9] H. Bais, M. Machkour, and L. Koutti, "Querying database using a universal natural language interface based on machine learning," Mar. 2016, doi: 10.1109/it4od.2016.7479304.

[10] L. Koutti, H. Bais, and M. Machkour, "An independent-domain natural language interface for multimodel databases," *International Journal of Computational Intelligence Studies*, vol. 8, no. 3, p. 206, 2019, doi: 10.1504/ijcistudies.2019.10024284.

[11] A. Mukherjee and U. Garain, "A review of methods for automatic understanding of natural language mathematical problems," *Artificial Intelligence Review*, vol. 29, no. 2, pp. 93–122, Apr. 2008, doi: 10.1007/s10462-009-9110-0.

[12] A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates, "Modern natural language interfaces to databases: composing statistical parsing with semantic tractability," 2004, doi: 10.3115/1220355.1220376.

[13] L. R. Tang, "Using a machine learning approach for building natural language interfaces for databases: application of advanced techniques in inductive logic programming," *Journal of Computer Science, Informatics and Electrical Engineering*, vol. 2, no. 1, pp. 140–60, 2008.

[14] J. Sànchez-Ferreres, J. Carmona, and L. Padró, "Aligning textual and graphical descriptions of processes through ILP techniques," in *Lecture Notes in Computer Science*, Springer International Publishing, 2017, pp. 413–427.

[15] M. Eineborg and N. Lindberg, "ILP in part-of-speech tagging — an overview," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2000, pp. 157–169.

[16] K. Shaalan and A. H. Hossny, "Automatic rule induction in Arabic to English machine translation framework," in *Natural Language Processing*, John Benjamins Publishing Company, 2012, pp. 135–154.

[17] D. Kazakov and S. Manandhar, "Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming," *Machine Learning*, vol. 43, pp. 121–162, 2001.

[18] J. M. Zelle and R. J. Mooney, "Learning to parse database queries using inductive logic programming," *Proceedings of the national conference on artificial intelligence*, pp. 1050–1055, 1996.

[19] S. Muggleton, "Inductive logic programming: issues, results and the challenge of learning language in logic," *Artificial Intelligence*, vol. 114, no. 1–2, pp. 283–296, Oct. 1999, doi: 10.1016/s0004-3702(99)00067-3.

[20] O. Al-Harbi, S. Jusoh, and N. Norwawi, "Handling ambiguity problems of natural language interface for questt tion ion answering answering answering answering," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 17, 2012.

[21] E. A. and O. D.O, "An algorithm for solving natural language query execution problems on relational databases," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 10, 2012, doi: 10.14569/ijacsa.2012.031027.

[22] D. Kazakov, "Achievements and prospects of learning word morphology with inductive logic programming," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2000, pp. 89–109.

[23] M. E. Califf and R. J. Mooney, "Advantages of decision lists and implicit negatives in inductive logic programming," *New Generation Computing*, vol. 16, no. 3, pp. 263–281, Sep. 1998, doi: 10.1007/bf03037482.

[24] G. Sazandrishvili, "Asset tokenization in plain English," *Journal of Corporate Accounting &amp; Finance*, vol. 31, no. 2, pp. 68–73, Nov. 2019, doi: 10.1002/jcaf.22432.

[25] J. Tiedemann, "Word to word alignment strategies," in *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, 2004, doi: 10.3115/1220355.1220386.

BIOGRAPHIES OF AUTHORS

Hanane Bais is a full professor of higher education, Moroccan School of Engineering (EMSI). She received her Ph.D. degree in Computer Science from the Ibn Zoher University, Agadir, Morocco in 2018, from Departments of Computer Science, Faculty of Science, University Ibn Zohr, Agadir, Morocco. Member of Laboratory LAMIGEP, EMSI Marrakech, Morocco. Her research interests include database system, natural language processing, and artificial intelligence. She can be contacted at email: h.bais@emsi.ma.

Mustapha Machkour is a full professor of higher education, Department of Computer Sciences, Head of the Intelligent Computing Models and Knowledge Engineering (M3IC) Team, Ibn Zohr University, Agadir, Morocco. Member of Laboratory of Computer Systems and Vision, Faculty of Science, Ibn Zohr University, Agadir, Morocco. Current research interests include natural language processing, database system, logic and artificial intelligence, and image processing. He can be contacted at email: machkour@hotmail.com.