XPathia: A Deep Learning Approach for Translating Natural Language Into XPath Queries for Non-Technical Users

Karam Ahkouk, Mustapha Machkour
Computer Systems and Vision Laboratory-Faculty of Sciences,
Ibn Zohr University, 80000 Agadir, Morocco

***Abstract*—XPath is a widely used language for navigating and extracting data from XML documents due to its simple syntax and powerful querying capabilities. However, non-technical users often struggle to retrieve the needed information from XML files, as they lack knowledge of XML structures and query languages like XPath. To address this challenge, we propose XPathia, a novel deep learning-based model that automatically translates natural language questions into corresponding XPath queries. Our approach employs supervised learning on an annotated XML dataset to learn accurate mappings between natural language and structured XPath expressions. We evaluate XPathia using two standard metrics: Component Matching (CM) and Exact Matching (EM). Experimental results demonstrate that XPathia achieves a state-of-the-art performance with an accuracy of 25.85% on the test set.**

***Keywords*—Deep learning; XML databases; neural networks; text-to-XPATH; natural language processing**

I. Introduction

The use of databases is the activities' base that the majority of companies rely on. Many types of databases exist in the world, amongst them there are Graph databases, NoSQL databases, Object-oriented databases, XML databases, Relational databases, etc. All this types of databases are based on containers called database management systems that help manipulate these databases and give accessibilities features for users and also for other applications.

Dealing with these systems is not an easy task for people who lack of knowledge on database management. Also the use of these systems requires advanced skills and enough expertise on some specific technical languages, for instance, the ignorance of some languages like XPath or XQuery for getting the relevant information form XML document in XML databases is considered as a big obstacle and can create limitations to get the searched data in a reasonable time.

An ideal solution would be to communicate with the database management systems using only the mother-tongue language; this can be done by using an intermediate interface that handles well organized natural language sentences and translate them to the target script which is in our case XPath. Xpath is one of the languages that help manipulate the XML documents included in XML databases in an easy and straightforward way. In contrast with the other languages, XPath has a very simplified syntax and direct structure; also the possibilities that it affords for extracting data from xml documents helps people get the searched information without complexities. Xpath is the one that can be used with minimal effort; its syntax that is based on Path concept provides 200 built-in functions with expressions and operators to extract XML nodes and their values.

The type of this interface enters in the category of solutions based on Natural Language Processing (NLP) tasks that process NL sentences in order to make a corresponding translation, doing actions, generating codes, etc. There are many sub-tasks of the NLP that deal with natural text transformation. For example, Text classification, Named entity recognition, Information retrieval, Text summarization, Text generation, Sentiment/emotion analysis are all sub tasks of NLP. The objective of all these solutions is the ability to handle the meaning from a raw text and transform it to another form that answers a particular problem. In our case the problem is similar to Text classification and Information retrieval since the goal is to classify the user natural language sentence and get the relevant information in order to build a target XPath query which will be used for extracting the wanted information from an XML document.

The focus in this work is to leverage the ability to interact with XML databases using only the natural language. In other words, the possibility to translate the sentences of users to a runnable and valid Xpath query with all its paths, nodes and expressions. In this paper, we present our solution that handles the problem of using the Natural Language as a source medium to interact with XML Databases. The proposed model is based on Deep learning with a Path-Expression based approach that treat the path, nodes and predicates in an independent manner in order to reduce errors. The work is evaluated against a new XML dataset that we provide publically on GitHub https://github.com/karamsa/XMLSet using two metrics, the CM (Component Matching) and EM (Exact Matching), and achieves a score of 25.85% on the test set of the XML corpus to be a new base for future models for the training and the evaluation.

A. Contribution

In this paper, we present 2 contributions to the task of translating Natural Language to Xpath queries for XML databases using Deep Learning and a Path-Expression based approach to make prediction of nodes, Expression, predicates, operators and values independently and without using the content of the XML Nodes. Our contribution to this area is two fold:

- We propose a new XML Dataset that contains up to 2176 pairs of natural language sentences with their XPath queries and annotations in a well structured and easy-to-use format. We provide this Dataset publicly on GitHub for future models for training and evaluation purposes.
- A new model (we called it Xpathia) that makes predictions of Xpath queries using deep learning and Path-Expression based approach. The model predicts each part of the target query independently using several sub-modules.

B. Outline

The reminder of this paper is structured as follows: The Section 2 presents the related works regarding the previously proposed solutions. Section 3 and 4 presents our contributions on the translation of Natural Language sentences to XPath. Section 5 includes the evaluations of the model. The section 6 discusses the experimental results and the final section contains the conclusions of our work.

II. Related Works

XML is a widely used markup language, valued for its simplicity in data extraction. Originally developed for large-scale electronic publishing ([1]), it is now applied in diverse areas such as data sharing, system communication, and database storage. Its flexible format supports both structured and unstructured content, and its platform- and language-independence have contributed to its broad adoption ([2]).

Recent research in XML has focused mainly on content for tasks like information retrieval and document clustering ([3]), often overlooking the structural aspects. This neglects a key feature of XML—its hierarchical structure and meaningful tag names—which are essential for accurate content interpretation ([4]; [5]). To improve retrieval and analysis, both content and structure should be utilized together ([6]; [7]).

Databases are central to most organizations, ranging from relational and NoSQL to XML and graph databases. These are managed through database management systems (DBMS) that enable access and manipulation of data. However, using such systems can be challenging for users without technical expertise. In particular, querying XML databases requires knowledge of XPath or XQuery—languages that many users find difficult to learn. This creates barriers to accessing information efficiently.

To address this, natural language interfaces (NLIs) have been proposed to translate user-friendly queries into structured query languages. In [8] the authors provides a foundational overview of NLIs, highlighting their potential to make database interaction more accessible, especially for non-experts.

While XML's self-descriptive design enables platform-agnostic data storage and transport, translating natural language (NL) directly into XML structures remains inherently challenging. NL, which is an unstructured language, is governed by grammar and contextual rules that are dedicated for communication purposes, whereas XML organizes data hierarchically for machine readability and data storage. This fundamental mismatch necessitates intermediate query languages like XQuery or XPath in order to create a bridge between Natural Language and XML. This intermediate language can play a pivotal role to remove the semantic as well as the structural gap.

The existing research addressing this challenge falls into two broad categories: the ones that are linguistic and rule-based and the others emerging deep learning methods, though the latter remains underexplored.

The linguistic rule-based approaches is the category of methods that rely on syntax parsing, part-of-speech tagging, and predefined rules to map NL to XML queries.

Tannier et al. ([9], [10]) utilized the INEX 2004 dataset to develop a system combining part-of-speech tagging, context-free grammar rules (e.g., NP for noun phrases), and discourse representation theory (DRT). By constructing discourse representation structures (DRS) from NL sentences and converting them into SQL-like queries, their approach retrieves XML nodes. However, it struggles with lengthy inputs, negations, and requires synonym dictionaries (e.g., mapping "employee" to "emp") to align NL terms with XML schemas.

Li et al. ([11]) introduced NaLIX, a natural language interface leveraging MINIPAR for syntactic parsing and iterative user feedback to refine XQuery outputs. While functional, NaLIX relies on manual rule engineering and interactive prompts, limiting its scalability.

To simplify query generation for non-expert users, Li et al. ([12]) proposed the Meaningful Query Focus (MQF) framework. MQF decouples translations from XML schema specifics, enabling cross-domain compatibility. However, it mandates exact tag names, necessitating synonym substitution layers to align user terminology with schemas.

Joseph et al. ([13]) expanded versatility by supporting multi-sentence queries, yes-no questions, and nested structures. Their system parses dependencies, replaces tokens with XQuery fragments, and uses WordNet ([14]) for synonym mapping. Despite its flexibility, it falters with unseen domains or terms absent from WordNet, underscoring reliance on predefined lexicons.

The hybrid and schema-agnostic approaches is the category of methods that include efforts to reduce dependency on rigid schemas or manual rules. Nassiri et al. ([15], [16]) designed a system to unify queries across data models (including XML) via syntax validation, tokenization, and intermediate universal query generation. While effective for structured inputs, it lacks NL compatibility and relies on manual engineering.

Gupta et al. ([17]) combined dependency parsing with Paninian grammar for semantic parsing, mapping NL chunks to domain models. Though innovative, their hybrid approach lacks adaptability for complex inputs without deep learning integration.

Dorrn et al. ([18]) emphasized data-driven methods, combining XML and NLP techniques for automatic information extraction from corpora. However, limited evaluations and reliance on mixed methodologies hinder practical scalability.

Based on semantic and for cross-Domain databases subhro et al. ([19]) proposed declarative rules to map NL descriptions

to formal queries (e.g., XPath), while Lopez et al. ([20]) developed PowerAqua, a tool translating NL into logical queries across heterogeneous semantic sources. PowerAqua aggregates answers without requiring schema knowledge but faces challenges in web-scale diversity.

XML is a flexible, structured language widely used for data storage and sharing, but querying it requires technical skills in XPath or XQuery. Existing solutions use rule-based or traditional methods to translate natural language into XML queries but often lack scalability and adaptability. Deep learning offers a promising alternative by reducing manual rule dependency and improving cross-domain performance.

III. XML Datasets

As we have previously, presented the base idea is the subdivision of the problem into multiple small problems without following a pre-defined sketch that still in the same time dissociates the tasks and operations done by the main model and create an independency between the output spaces, thing that is essential for a healthy prediction process.

The first problem encountered while working on this problem is the lack of databases dealing with this transformation type; thus creating a database for evaluating the model is mandatory in order to have a robust model that can generalize for unseen inputs. Bridging the natural language to Xpath was another obstacle for the model, since they both have different natures and purposes and Xpath is the intermediate language for extracting content portions from XML documents.

A. Existing Corpuses

The decision to build an XML dataset fully dedicated to answering questions in natural language with structured labels was based on the lack of similar corpuses that can be used for the evaluation of our model. This task was one of the harsh steps done prior the creation of the model and it is essential for a preferment system that must generalize not only to unseen inputs but also for unseen structures in the sets of the dataset.

From the literature, the use of Natural language to extract XML portions using structured languages wasn't deeply considered and has been treated as one of most difficult tasks in the NLP area.

This was mainly due to two reasons; first is the incompatibility of the two languages. Second is the lack of corpuses for the evaluation of the wanted model. In this context, the number of corpuses that were presented in this area in the recent decade was very limited. While these few datasets were created in the first place as a contribution to the task, in the other hand they bear lots of technical issues and difficulties that make their use such an impossible thing especially for the training and the evaluation of our model.

XML retrieval, the process of querying and extracting structured information from XML documents, relies heavily on specialized datasets to advance research, benchmark systems, and refine methodologies. These datasets are tailored to address diverse challenges in handling XML's hierarchical and semi-structured nature, offering varied use cases ranging from algorithmic stress-testing to domain-specific analyses. Among these datasets, the INEX Dataset ([21]) stands out as a foundational resource. Developed through the Initiative for the Evaluation of XML Retrieval, INEX provides standardized test collections that simulate some limited scenarios, enabling researchers to evaluate the precision, recall, and efficiency of XML retrieval algorithms. Its role in academic competitions and collaborative benchmarking has made it a popular dataset for comparing old systems that process structured queries, particularly in environments where granular content extraction—such as retrieving specific sections of documents—is not critical.

For performance-oriented evaluations, the XMark Benchmark ([23]) was proposed for this purpose. This synthetic dataset models an online auction platform, generating XML data with deeply nested hierarchies and scalable file sizes. Its design intentionally incorporates complexity to mimic real-world data structures that is useful for making stress-testing XML databases and query engines. Researchers use mainly use XMark to measure how systems handle large-scale data, complex XPath queries, and join operations, providing insights into scalability and optimization strategies. However that corpus is not intended for making natural language translation to XPath queries as it doesn't has this structure. Complementing this, the SIGMOD Record XML Dataset ([22]), which is another dataset for XML files with focuses on XML query processing techniques. Curated by the ACM Special Interest Group on Management of Data, this dataset supports experiments in query optimization, indexing, and retrieval efficiency, often serving as a testbed for novel methodologies in academic publications. While it is adequate for research targeting query optimization, this cannot be used for our case as there is no samples that handle the translation pairs.

Domain-specific datasets further enrich the landscape of XML retrieval research. The DBLP XML Dataset ([24]), for instance, offers a comprehensive corpus of computer science publication metadata. Covering millions of journal articles, conference papers, and academic references, it enables studies on bibliographic data retrieval, citation analysis, and trends in scholarly communication. Its structured yet semantically rich content makes it invaluable for exploring hybrid approaches that combine XML retrieval with bibliometric analytics. From one hand, it can be seen as a challenging dataset that put together an important volume of XML files, however, it is structure does not involve the natural language questions that are required for handling the task by our model, and thus it is unusable for this kind of training or evaluation. In the same context, the domain specific nature of the corpus creates an obstacle for the model to generalize for unseen domain which is an unwanted behavior for our model.

Similarly, the IMDB XML Dataset ([25]), derived from the Internet Movie Database, provides a heterogeneous collection of entertainment industry data. This includes detailed records on films, TV series, actors, directors, and production crews, offering researchers a platform to tackle challenges in querying diverse entity types, managing cross-linked data, and extracting insights from semi-structured multimedia metadata. Despite having a corpus that contains huge volume of domain specific data structured in XML files, these latter can't be transformed to a structured question-answer dataset to be eligible for the training and evaluation not only for our model but also for future models dealing with the process of translation.

Together, these datasets cannot be used for distinct facets of NLP using deep learning. Benchmarks like INEX and XMark prioritize technical rigor, enabling apples-to-apples comparisons of system performance, while domain-specific resources like DBLP and IMDB facilitate applied research in real-world contexts. For a granular comparison of these datasets—including their scope, structural complexity, and limitations —Table I synthesizes key attributes with a detailed comparison of their features.

TABLE I. AN OVERVIEW OF THE EXISTING DATASETS

Dataset	Cross Domain	Number of Elements	Max-depth	Avg-depth	Limitations
INEX	Yes	12107	-	6.9	irregular tagging, incomplete schemas
SIGMOD Record XML	No	11526	6	5.14107	less useful for cross-domain XML retrieval
XMark Benchmark	No	Depends on configs	Depends on configs	Depends on configs	Artificially generated hierarchies
DBLP XML Dataset	No	3 000 000	5-7	3.5	Not usable for semantic search or xml retrieval
IMDB XML dataset	No	Variable	Variable	Variable	Not usable for semantic search or xml retrieval

For the purpose of compatibility and since all the previous datasets presented in this paper don't offer a base for NLP tasks as well as the lack of XPath as a target language for the translation, we decided to build a new dataset (we called it XMLSet) that can encompasses all the missing things in one corpus and could be used by future models either for validation or training purposes.

B. XMLSet

The corpus is built based on different xml sources and underwent series of steps in order to have a usable dataset in the context of NLP. First, the xml databases have been cleaned and preprocessed automatically in order to remove non-sense names and replace non significant names with the ones that have signification. Second and since all XML databases are collected from existing source from internet as shown in https://github.com/karamsa/XMLSet, we have converted some of the xml sheets to XML databases in order to leverage our dataset with data from real word that is crucial for making a model that can generalize to unseen data; also we have kept the nature of some of the xml data that contain more than one XML root. We consider this as a necessary noise that makes the model more robust for real usage. For the aim of having a cross-domain dataset, we made sure that our new corpus includes a variety of domains from bookstores, purchases, plants, food, movies, journals, and more than 28 distinct topic; the table II shows a general comparison between the existing datasets and the XMLSet created by us. Since the XMLSet differs totally from the existing corpuses, we only show shared properties' statistics, such as the depth of xml nodes, average dept, and number of elements in the dataset and so on.

TABLE II. XMLSET AGAINST THE EXISTING DATASETS

Dataset	Cross Domain	Number of Elements	Max-depth	Avg-depth
INEX	Yes	12107	-	6.9
SIGMOD Record XML	No	11526	6	5.14107
XMark Benchmark	No	Depends on configs	Depends on configs	Depends on configs
DBLP XML Dataset	No	3 000 000	5-7	3.5
IMDB XML dataset	No	Variable	Variable	Variable
XMLSet	**Yes**	**37,788**	**8**	4

The construction of the XMLSet dataset was a prior step in the process of making a model that deals with NL to XML through XPath as an intermediate language. In contrast with the existing datasets that are not dedicated to NLP processing tasks, the XMLSet is the first corpus that can be used for models that rely on deep learning. It contains more than 2176 pairs in form of a natural language sentence and an answer in XPath language. The dataset is also leveraged with a ground truth section for each pair that describes the answer in a structured way making it adequate not only for deep learning processing but also for any XML retrieval form and can be used by models for training and evaluation purposes. The Figure 1 shows an example of a pair from XMLSet including the details about the labels sections (ground truth).

```
{
  "question": "find all addresses but the first two ones",
  "query_id": 2026,
  "xpath": "//address[position()>2]",
  "ground_truth": {
    "slots": [
      {
        "root_type": "//",
        "node": "address",
        "source": "position()",
        "op": ">",
        "value": "2"
      }
    ]
  },
  "schema_index": 3
},
```


Fig. 1. An Example of a natural language question and its answer including the details about the labels sections from XMLSet dataset.

Regarding the pairs statistics, the dataset is built on top of 30 separate XML database files. Some of them are bigger than the others especially in terms of number of elements and this is particularly helpful to ensure a diverse size across the whole corpus that is divided onto 2 sets. The data split of XMLSet is as shown in table III:

In terms of difficulty, we made sure that XMLSet contains a diversified pairs with easy, moderate and difficult queries;

TABLE III. THE DATA SPLIT OF XMLSET

	Train	Test
#Pairs	1472	704
#Unique NLs	1255	673
#Databases	30	30

The table IV shows a detailed analysis of the XMLSet.

TABLE IV. XMLSET'S DETAILED STATISTICS

Dataset	# Relative Paths	# Multi-paths	# Predicates	# Functions
XMLSet	77	**743**	**770**	**324**

In XPath, the difficulty of a query can be measured using different criteria. First we find the number of the nested paths which reflects the number of nodes to be included in the final XPath query. Then comes the number of predicates which has a direct link to the targeted nodes in the XML file. Another element that adds difficulty to XPath queries is the inclusion of functions in the predicates; thus is considered as a double difficulty as it puts together the predicates and functions in one place. Relative paths, that include the '..' are also a factor of complexity in XPath queries; thus adding it to some pairs in the dataset challenge the model on those complex transformations.

In the same context, we built XMLSet to follow the question-answer paradigm; hence not all queries return results as we want to have a system that can work in real situations where a query entered by a user might have no corresponding answer in a database.

IV. THE MODEL

A. Overview

Exploiting XML files using natural language solely, requires an intermediate language that can manage the translation operation by bridging the nature of each language; thus the use of a structured language such as XPath is mandatory for this process. The use of XPath comes from multiple factors. First, its simplicity; XPath is a simple language that relies on paths. It provides the possibility to extract nodes from XML files using a short syntax. Additionally, XPath can be applied directly to XML documents in comparison to other languages like XQuery which requires to be re interpreted. Another reason for choosing XPath is the richness in terms of functions; it provides more than 200 useful functions that can be used along with inherent expressions to exploit XML files effectively.

The conversion of NL sentences into XPath is a thorough task that should take in consideration the meaning inherent in the user sentence. To this end and since the two languages are different in nature, we decided to simplify the problem by adopting an straightforward mapping.

The model should predict the nodes and the predicates to be included in the final XPath query. With nodes the subset of the following items:

$$\text{N=[T; ”null”, ”.”, ”..”, ”*”, ”1”, ”last()”, ”position()”]} \quad (1)$$

In addition to T (the tags from the XML document) we also add to the output space the relative paths '.' and '..', '*' for any tag in the document, '1' and 'last()' for the first and last node and finally the 'position()' that helps select nodes based on a given position.

In the other side, a predicate P is a structure that holds multiple elements: A node from the previous list N, an operator from the set of possible operators O and a value from the list of values V mentioned in the user input.

$$\text{P = [N(i), O(l), V(t)]} \quad (2)$$

We define O as the list of supported operators in the XPath syntax as following:

$$\text{O = [”null”, ”=”, ”<”, ”>”, ”<=”, ”>=”, ”!=”, ”+”, ”-”]} \quad (3)$$

Using those elements and by exploiting the incorporate structure of XPath, we reformulate the problem and we simplify it further by limiting the output space in each item of the target query.

The problem then can be summarized as the prediction of one or multiple paths of nodes and predicates in the following format:

$$Q = //N_1P_1/N_2P_2.../N_nP_n \quad (4)$$

The paths are separated by delimiters of two types; the '//' which refers to selecting elements in the XML document from no matter where they are and that correspond to the current selection in the path. The second type is '/' which provides the possibility to select nodes depending on the root path or the previous one.

For example, if we take an XML file that stores the orders done by clients, the following XPath query will find the last saved order in the document:

$$//orders[last()] \quad (5)$$

We notice that the query contains only one path that includes the 'order' node and a predicate which contains a node as a function 'last()'. The operator and the value for the predicate in this case are optional. This query can take another form. For example selecting the last but one order saved in the XML document:

$$//orders[last() - 1] \quad (6)$$

In this case, the predicate contains a node as a function 'last()', the operator '-' and the value '1'. The table V shows examples of XPath queries with their descriptions.

TABLE V. SOME XPATH QUERIES AND THEIR DESCRIPTION

Query	Description
//orders[last()]	Get the last order wherever it is.
//orders[last()-1]	Get the last order but one from all order in the XML database.
//categories/description	Get the description node of categories nodes.
//cd[year=1985]	get the cds that have been published in 1985.
//company/employee[salary¿5000]	get companyies' employees that have a salary higher than 5000.
//company/employee[1]/*	get all information of the first employee of each company.

The same pattern can be used to represent complex XPath queries that involve different nodes, conditions, and multiple paths in the same XPath request. Hence an XPath query can be split into different parts; each one will be part of a specific predication process independently to the others.

B. Model Details

The goal of this work is the creation of a model that can generate XPath queries effectively using the learned patterns in the training dataset of XMLSet.

Since the translation of the natural sentence to XPath is a problem that can be reformulated as the prediction of one or multiple paths separated by roots and while each path can be represented by a node and a predicated as presented in the previous section, we can then use this powerful structure of XPath to make prediction of the target query in an effective manner that can help the model generate to unseen inputs from users.

We split the problem into several sub-problems in a way that dissociates the inputs and the outputs of each element in the prediction process. In this context and based on an input from the user, the natural language sentence as well as the nodes of the XML file are fed to the model that run a series of prediction operations in order to construct a valid query that can be executed against an XML engine, so the result is returned to the user as shown in figure 2 that displays the architecture of our model (XPathia) in details.

The natural language sentence is tokenized and the XML file is preprocessed to get the XML nodes that make the structure's file then the two inputs are concatenated each other and the Meta data are added. We leverage the combined inputs with separators to distinguish between the user input and the XML datafile. We use [SEP] from the BERT pretrained model [26] to separate the natural language sentence form the structure of the XML file and also for separating each tag automatically extracted from the XML document. The BERT (Bidirectional Encoder Representations from Transformers) is a language representation model that provides pertinent contextualized word embedding in contrast to static one proposed by other models. In addition to the contextualized word embeddings that BERT provides using representation from left and right layers of the transformers model [27], we also leverage our model

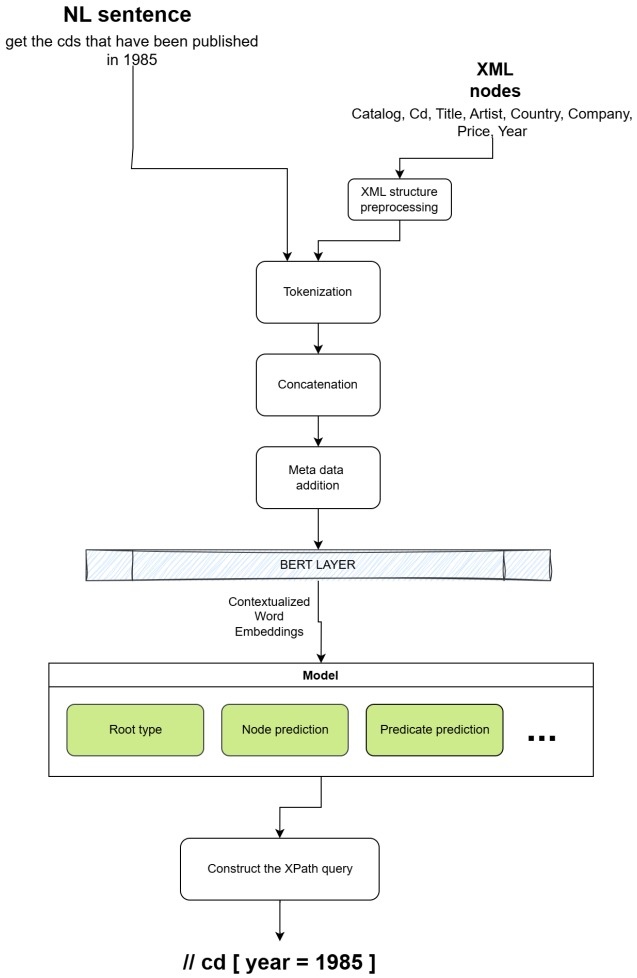

Fig. 2. The Architecture of our system.

with the [SEP] token as a separator as well as the special token [CLS] that holds the whole meaning of the input in one word embeddings. This token can concentrate the representation from both sides as BERT is a bidirectional model. These metadata are essential to get focused representations that will be used in the next step. The query generation process takes place the BERT layer is completed. Our model is composed of several sub modules and each sub-module is dedicated to generate specific elements of the query. We don't use a sketch filling approach as in our case the whole query will be predicted and this will be done once not sequentially. Since the redefined structure of an XPath query is more straightforward as have been shown in the previous section, we define six sub-modules for generating the target XPath query as shown in figure 3.

For each sub-module, the sentence S and the nodes N will be fed as I:

$$I=[[CLS];S;[SEP];n_1;[SEP];n_2... \ [SEP]n_z] \quad (7)$$

The first step consists of predicting the number of paths that will construct the final XPath query. In contrast to what we have done in other works; we are not choosing a template from

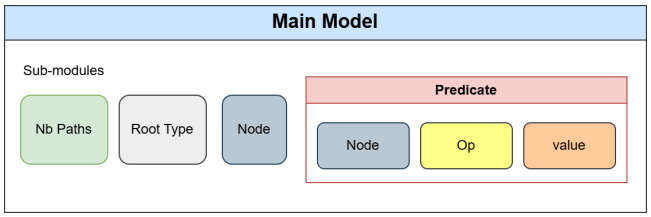

Fig. 3. The Main model and the included sub-modules for the generation process.

a predefined set of possible structures, but we will generate the query structure using a dedicated sub-module. This lets the generation of a non predefined structure possible.

$$P_n(n|I) = softmax(X_n tanh(Y_n EMB_{[\text{CLS}]}))_i \quad (8)$$

we use X and Y as two matrices with separate trainable parameters that help predict the number of paths in the final query. This sub-module is based on the use of the attention mechanism [28] and the work done by sqlnet (2017)). The softmax function normalizes the scores of all probabilities and the top one is selected as the final number of paths in the target XPath query.

As a path in XPath always starts either by '//' or by '/', the root type module is triggered to predict the root types of all the paths in the final query. Using the predefined list of root types ['/', '//'] of each path we put:

$$\begin{aligned} r &= V_{RootType} tanh(W_{RootType} EMB_{[\text{CLS}]}) \\ P_{RootType_{(i)}}(RootType|I) &= softmax(r) \end{aligned} \quad (9)$$

with $\text{EMB}_{[\text{CLS}]}$ the contextualized embedding of the whole input, $\text{V}_{\text{RootType}}$ and $\text{W}_{\text{RootType}}$ are two matrices with isolated trainable parameters fully dedicated to this sub-module. The softmax function normalizes the probabilities for the root types over each path and the highest ones are chosen for each step. In real work almost all XPath queries starts with '//'; hence a great deal of generated root types are '//' for the first path.

The node sub-module is the one in charge of predicting tags that will be included in each path. The tags are predicted based on the list of the fed XML nodes in the first step and using the meaning extracted from the natural language sentence of the user. A node in a specific path can be computed as following:

$$\begin{aligned} z &= S_{node} tanh(T_{node} R_{[CLS]/node(i)}) \\ P_{node_{(i)}}(node|I) &= softmax(z) \end{aligned} \quad (10)$$

with $\text{EMB}_{[\text{CLS}]/\text{node(i)}}$ the representation of a node i in the list of provided nodes and based on the special token CLS that involve the whole representation of the input; this way we enforce the meaning and concentrate the representation for each node extracted from the xml document . S_{node} and W_{node} are two matrices with isolated trainable parameters fully dedicated to this sub-module. The softmax function normalizes the probabilities for the xml nodes over each path and the top candidates are chosen to be predicted as outputs.

Regarding the predicates and since they are composed of three sub-modules (Node, Operator and Value) we decided to use the same structure of modules as before. For instance, the node sub-module within the predicate has the same structure as the one in the path. Both they generate a node based on the list of xml tags from the input and based on the [CLS] token, however the latter is shown before the predicate while the other is inside the predicate. While the two sub-modules use the same structure, they differ in the parameters. We use independent matrices for each sub-module even if they have the same structure; this can be understood as we want to dissociate the training of each module. The node of a predicate can be computed as following:

$$\begin{aligned} x &= M_{node_{predicate}} tanh(N_{node_{predicate}} R_{[CLS]/node(i)}) \\ P_{node_{predicate_{(i)}}}(node_{predicate}|I) &= softmax(x) \end{aligned} \quad (11)$$

The matrices $\text{M}_{\text{node}_{\text{predicate}}}$ and $\text{N}_{\text{node}_{\text{predicate}}}$ contain independent trainable parameters exclusively dedicated to this sub-module and not shared with the others. The softmax function normalizes the probabilities of XML nodes along each path, selecting the top candidates for prediction.

For the operator inside the predicate, we use two separate matrices and the representation of the whole input given by the user. For each path we select one item from the list of possible operators ['null', '=', '¡', '¿', '¡=',' ¿=', '!=', '+', '-'] and we compute:

$$\begin{aligned} q &= E_{Op} tanh(F_{Op} EMB_{[\text{CLS}]}) \\ P_{Op_{(i)}}(Op|I) &= softmax(q) \end{aligned} \quad (12)$$

With E_{Op} and F_{Op} two matrices with trainable parameters specific for this module and will predict an operator for a given path in the target XPath query. The probabilities for each Op(i) are calculated and the top operators are selected for the predicates using the softmax function.

The 'null' element in the list of operators refers to when there is no operator for the predicate; thus the 'null' item will be predicted as the top candidate with the highest probability. In the same context, we leverage all lists with a 'null' value that can be predicted when there is no item to be predicted. For instance, the query bellow doesn't include operator nor a value in the predicate, hence the item 'null' will be predicted for both positions:

$$//messages[1] \quad (13)$$

This gives us the flexibility to stick to the simplified structure of XPath and also run the same number of modules regardless the number of paths and without taking the burden of the structure in the target query.

$$//messages[1, \textbf{null}, \textbf{null}] \quad (14)$$

V. Evaluation Results

Our model (XPathia) is trained using XMLSet dataset that we have presented in previous sections. The training is made on windows OS with a 4G in GPU for the training set of the dataset and for several epochs. The chosen batch size is 4 and we use [29] for the training with default values for the learning and dropout rate, respectively 1e-3 and 0.3. The training process (Figure 4) starts stabilizing after the nineteenth epoch and the accuracy on the training set achieves a score of 29%.

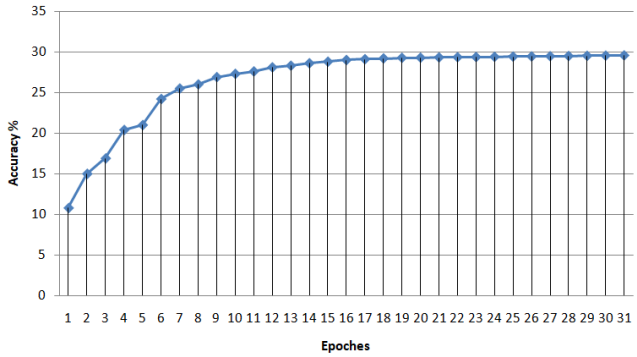

Fig. 4. The Training score on for our model XPathia.

All the sub-modules that composes our model XPathia are trained separately on the pairs of XMLSet that are included in the training set and evaluated against the test set only once.

As we have proposed in our previous work [30] and [31], we use two metrics to evaluate our model. The first metric is the component matching. This metric measures the correctness of outputs of each sub-module and provides an accuracy score on how many parts of the XPath queries have been predicted correctly. Since the XMLSet provides a structured ground truth, the matching is straightforward based on a set matching between the sub-module output and the corresponding item on the ground truth. The loss scores are summed in order to optimize the parameters hence giving better prediction and minimizing the loss score for each sub-module. The table VI shows the performance of XPathia model on the test set of XMLSet dataset using the component matching metric.

TABLE VI. The Performance of XPathia Model on the Test Set of XMLSet Dataset Using the Component Matching Metric

Model	# NB paths	Root types	Node	Predicate		
XPathia	98.86%	66.4%	64.2%	**40.1%**		
				Node	Operator	Value
XPathia	**98.86%**	**66.4%**	**64.2%**	**41.9%**	**91.61%**	**49.00%**

With the 'nb paths', the accuracy of the module that predict the number of steps in the final XPath query. XPathia achieves a score of 66.4% on the prediction of the root types as well as 64.2% of correct answers on the test set for the prediction of the nodes that will construct each path in the query. The accuracy score for the predicates is 40.1% of the correct answers generate by the corresponding sub-module.

Another metric used to evaluate our model XPathia, is the total matching or what we called in our previous works Exact matching. This metric helps measure the correspondency of a generated query by the components of the main model with the ground truth query. The whole output is considered as correct when all parts match exactly all items in the ground truth data.

The table VII shows the performance of XPathia when measured with the Exact matching on the XMLSet dataset.

TABLE VII. The Performance of XPathia When Measured With the Exact Matching on the XMLSet Dataset

Model	Train set (last epoch)	Test set
XPathia	29.0%	25.85%
XPathia-wo-values	**42.8%**	**36.05%**

The model achieves a score of 25.85% of correct queries on all pairs seen in the test set of XMLSet while it achieves a total score of 29.0% on the training set when trained for several epochs. We also explore the evaluation without taking in consideration the generated outputs related to values in the final query. When evaluated without taking in consideration the values, it achieves a score of 42.8% on the training set of XMLSet. **XPathia-wo-values** outperforms the base model by 10.2% on the test set and achieves an accuracy score of 36.05% which becomes the state of the art for the task of translating natural language to XPath for the aim of XML extraction.

VI. Discussion

The evaluation of the existing XML datasets in this work leads to several conclusions:

- The existing Cross-domain datasets are not suitable for XML retrieval using deep learning.
- Some of the datasets are configurable in size and complexity, useful for performance benchmarking but not cross-domain, with randomly generated data.
- Datasets with Large bibliographic corpuses are good for XML data management but not suitable for translation model training.
- The presented corpuses with Variable size and complexity are adaptable for research but not cross-domain.

Given these insights, and since there is no available dataset that can fill the lack of corpuses dedicated to NLP and deep learning, we have proposed XMLSet dataset with all advantages for the training and the evaluation of deep leaning based models. We emphasize on the quality of the corpus in terms of the size as well as the diversification of pairs with a wide coverage in topics like books, products, orders, universities, and more. This dataset is pairing natural language questions with corresponding XPath queries in JSON format and is split into train and test sets. While the XMLSet is a good starting point for future models, we believe that the dataset can be enhanced more by adding more items, creating more complex structures and might be leverage by an intermediate set for fine

tuning the models in the training process. Unlike previous work using XQuery, we introduce XPath to simplify translation, treating it as progressive generation process. We apply this approach based on [30] but with major differences in system architecture, model composition, and problem formulation. We choose a unified XPath structure for any query that can be generated by our model XPathia, and this helped us to reformulate the problem from a complex task to a relatively straightforward one. In comparison to other works that deal with the problem in a totally classical way, we choose to apply deep leaning to our model. The structure of sub-modules plays a pivotal role in the generation process as it provides a total independence between the output spaces. Each sub-module can learn to generate one type of inputs and this helps it enhance the quality of the generated elements as they have separate trainable parameters. In the same context and since all sub-modules parameters are isolated from each other, the errors are not propagated between the models, and thus, having an error in one sub-module doesn't affect the rest of the sub-modules. In relation to the training and as each element is predicted separately from the others, we notice a quick training process. The leaning starts stabilizing after few epochs; this can be understood as the model updates only the faulty parameters of the sub-modules with errors not the whole model, hence the back propagation is limited.

The model XPathia achieves a score of 25.85% on the challenging dataset XMLSet and 36.05% when we omit the values prediction, thing that makes it the new state of the art model. While the accuracy is relatively high for such a task, we believe the model can be more enhanced with exploration of new techniques that can be applied to the model in order to improve the prediction accuracy. We understand that the major improvement can come from the use of LLM models that can have a positive impact to our model. Also we believe that some processing to the input before the starting of the prediction can also have a good impact to XPathia.

VII. Conclusion

In this study, we introduced a new deep learning-based approach for translating natural language into XML queries through an intermediate XPath representation. Our key theoretical contributions include the creation of XMLSet, the first annotated dataset designed for training and evaluating deep models on this task across diverse domains, and the development of XPathia, a novel model that leverages a unified XPath structure for improved generalization. XPathia achieves a 25.85% accuracy on the XMLSet test set, establishing a new benchmark in this domain. Despite these advances, the model still faces limitations in handling complex or nested queries and in generalizing to unseen schema variations. Future research will focus on enhancing the model's ability to understand deeper semantic structures and expanding the dataset with more complex query patterns. This work contributes to the field by introducing scalable, data-driven methods that reduce reliance on manual rule engineering and make XML data querying more accessible to non-technical users.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] F. Ravat, O. Teste, R. Tournier, and G. Zurfluh, "Finding an application-appropriate model for XML data warehouses", Information Systems, Vol. 35, No. 6, 2010, pp. 662-687.

[2] E. Wilde, and R. J. Glushko, "XML fever", Queue, Vol. 6, No. 6, 2008, pp. 46-53.

[3] Samadi, N., & Ravana, S. D. "Xml Clustering Framework Based On Document Content And Structure In A Heterogeneous Digital Library". Malaysian Journal of Computer Science. 2023. 36(2), 124–147.

[4] A. Algergawy, M. Mesiti, R. Nayak, and G. Saake, "XML data clustering: An overview", ACM Computing Surveys (CSUR), Vol. 43, No. 4, 2011, pp. 25.

[5] H. M. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, Intelligent search on XML data: applications, languages, models, implementations, and benchmarks. Springer Science & Business Media, 2003, pp. 59-75.

[6] E. Asghari, and M. KeyvanPour, "XML document clustering: techniques and challenges", Artificial Intelligence Review, Vol. 43, No. 3, 2015, pp. 417-436.

[7] A. Tagarelli, and S. Greco, "Semantic clustering of XML documents", ACM Transactions on Information Systems (TOIS), Vol. 28, No. 1, 2010, pp. 3.

[8] Androutsopoulos I, Ritchie GD, Thanisch P. Natural language interfaces to databases – an introduction. Natural Language Engineering. 1995;1(1):29-81.

[9] Tannier, X., Girardot, JJ., Mathieu, M. Analysing natural language queries at INEX 2004. Advances in XML Information Retrieval. INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds) Advances in XML Information Retrieval. INEX 2004. (2005).

[10] Tannier, X., Geva, S. (2005). XML Retrieval with a Natural Language Interface. In: Consens, M., Navarro, G. (eds) String Processing and Information Retrieval. SPIRE 2005. Lecture Notes in Computer Science, vol 3772. Springer, Berlin, Heidelberg.

[11] Li, Y., Yang, H., Jagadish, H.V. (2005). NaLIX: an interactive natural language interface for querying XML. ACM SIGMOD Conference.

[12] Li, Y., Yu, C. Jagadish, H.V. Enabling Schema-Free XQuery with meaningful query focus. The VLDB Journal 17, 355–377 (2008).

[13] J. Joseph, J. R. Panicker and Meera M, "An efficient natural language interface to XML database," 2016 International Conference on Information Science (ICIS), Kochi, India, 2016, pp. 207-212

[14] George A. Miller. 1995. WordNet: a lexical database for English. Commun. ACM 38, 11 (Nov. 1995), 39–41.

[15] Hassana Nassiri, Mustapha Machkour, Mohamed Hachimi, Integrating XML and Relational Data, Procedia Computer Science, Volume 110, 2017, Pages 422-427, ISSN 1877-0509,

[16] Hassana Nassiri, Mustapha Machkour, Mohamed Hachimi, One query to retrieve XML and Relational Data, Procedia Computer Science, Volume 134, 2018, Pages 340-345, ISSN 1877-0509,

[17] Abhijeet Gupta, Arjun Akula, Deepak Malladi, Puneeth Kukkadapu, Vinay Ainavolu, Rajeev Sangal. A Novel Approach Towards Building a Portable NLIDB System Using the Computational Paninian Grammar Framework. International Conference on Asian Language Processing, 2012.

[18] T. Dorrn, N. Dambier, A. Müller and A. Kuwertz, "A Textual Information Extraction Application based on XML Data Models and a Multidimensional Natural Language Processing Pipeline Approach," 2022 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Biarritz, France, 2022, pp. 1-6.

[19] Subhro Roy and Dan Roth. 2018. Mapping to Declarative Knowledge for Word Problem Solving. Transactions of the Association for Computational Linguistics, 6:159–172.

[20] Lopez, V., Motta, E., Uren, V. (2006). PowerAqua: Fishing the Semantic Web. In: Sure, Y., Domingue, J. (eds) The Semantic Web: Research and Applications. ESWC 2006. Lecture Notes in Computer Science, vol 4011. Springer, Berlin, Heidelberg.

[21] Kazai, G. INitiative for the Evaluation of XML Retrieval. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA.

[22] SIGMOD Record. Index of articles from SIGMOD Record University of washington https://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html

[23] Albrecht Schmidt, Florian Waas, Martin Kersten, Michael J. Carey, Ioana Manolescu, and Ralph Busse. 2002. XMark: a benchmark for XML data management. In Proceedings of the 28th international conference on Very Large Data Bases (VLDB '02). VLDB Endowment, 974–985.

[24] Moreira, Catarina et al. "Learning to rank academic experts in the DBLP dataset." Expert Systems 32 (2015): 477 - 493.

[25] IMDb Non-Commercial Datasets. 2024. https://developer.imdb.com/non-commercial-datasets/

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding arXiv:1810.04805, 2018

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. arXiv:1706.03762, 2017

[28] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate* arXiv:1409.0473, 2015.

[29] Diederik P. Kingma, Jimmy Ba. *Adam: A Method for Stochastic Optimization* arXiv:1412.6980, 2014

[30] Karam Ahkouk, Mustapha Machkour, Khadija Majhadi, Rachid Mama. 'SQLSketch: Generating SQL Queries using a sketch-based approach'. Journal of intelligent and fuzzy systems . 1 Jan. 2021.

[31] Ahkouk, K., Machkour, M. SQLSketch-TVC: Type, value and compatibility based approach for SQL queries. Appl Intell 53, 3889–3898 (2023). https://doi.org/10.1007/s10489-022-03587-0