Automatic Database Schema Generator

Sayali Sant¹, Amruthkala Bhat², Neha Tiwari³, Purva Raut⁴

¹²³Student, Department of Information Technology, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India
⁴Assistant Professor, Department of Information Technology, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India

Abstract - Automatic Database Schema Generator is a tool that facilitates schema designing from Natural language based textual requirements as an input from the user, thus, automating the process of extracting probable entities and their attributes, identifying primary and foreign keys, etc. and thereby, eliminating time consuming requirement analysis phase in the project development lifecycle. This paper proposes how Natural language based textual requirements can be analysed to extract the entities from plain text and produce a database schema. Natural language text can tend to be ambiguous due to varied contextual meanings associated with the words. In order to reduce this ambiguity, the system makes use of Domain Ontology to identify associated terms in the given context, thus increasing the efficiency and richness of the database hence created.

Key Words: Domain ontology, Natural Language Processing, Schema, Entities, Attributes.

1. INTRODUCTION

A majority of projects in the industry make use of user data or other related information that needs to be stored in the application systems for further use. A large amount of such information might be later used by the application system itself for further processes or by the service provider. All these linked data are saved in application databases for later use. Therefore, designing a database that includes majority of useful information and that does not save redundant or unimportant data is crucial. Currently this process is carried out manually by repeated analysis of client requirements and multiple iterations of validation from the clients. A major chunk of software development time and efforts are to be invested in the initial phases of the cycle for the project to be a success. It is important that the fundamental idea of the project cycle be clear and strong enough. Any loophole in this phase will lead to a cascading effect in further stages. Generating the Database Schema for the project from user specifications involves multiple iterations of requirement analysis and rigorous client communications for validation. Projects of all scales make use of this approach irrespective of the type of solution or repetition of business uses. Rapid Application Development processes involve short spanned cycles, where schema generation from analysis result is important, failure of which may affect the timely deployment of the product.

The system aims to create an automated environment for analysing textual user requirements and formulating a relevant and client-domain specific database schema by extracting appropriate entities, associated attributes from the analysed text using Domain Ontology. The entities are then mapped according to their relations and key attribute values are identified to compose a full-fledged Relational Schema for the end user.

2. RELATED WORK

2.1. Automatic generation of schema from nested key-value data

This system automatically transforms self-describing, nested key-value data formats such as JSON, commonly found in NoSQL systems into traditional relational data that can be stored in standard RDBMS. This process includes a schema generation algorithm that discovers relationships across attributes of deormalized datasets to organize them into relational tables. The next process includes a matching algorithm to identify attributes with overlapping entities to merge them together under one entity to reduce data repetition. This system is most useful in cases where databases need to be propagated from a NoSQL system to a relational system thus helping users gain semantic understanding of complex, nested datasets.

Figure -1: JSON conversion to RDBMS format

Performance:

For a large Twitter dataset, the three consecutive phases of the process required 31 hours, 3.6 hours, 31 minutes respectively. In the second phase, it was observed that some
attribute matches were semantically related; however, a large number of matches were not semantically related and should not have been matched.

2.2. Automatic Relational Schema Extraction from Natural Language Requirements Specification Text

[1] This methodology deals with automatic construction of the relational database schema by identifying the key attributes from SRS using “rule-based approach”. The system architecture and working of each module in the system is as below:

Input: The system takes natural language textual requirements as input.

Domain Knowledge Elicitor: [1] This module splits the SRS into sentences and tags each word from all the sentences as nouns, verbs, adjectives, etc. using POS (Parts of Speech) tagging. Tagging of words is necessary to chunk the words that form noun phrases or verb phrases. The phrases are classified using simple phrasal grammar.

Schema Generator: This module identifies entities, attributes, methods and relationships based on simple rule-based approach from S-V-O pattern: Translating Nouns to Entities, Translating Noun-Noun to entity property according to the position, Translating the lexical verb of a non-personal noun to a method of this noun, Translating S-V-O structure to a class diagram with Subject and Object as Entity and verb as relation.

Identification of Primary Key (PK) and Foreign Key (FK): [1] A rule based approach is used to identify the primary key attribute from the attributes of all the tables.

3. PROPOSED APPROACH TO BUILD

Now we discuss the detailed approach of our proposed system to extract entities and their respective attributes from natural language requirements. It includes mainly following modules. [2] They are accepting the text input and tokenizing it followed by their POS tagging, parsing of ontology represented in OWL, identification of entities and attributes and finally, extraction of key attributes [1].

First module outputs the tagged text using a “Parts of Speech” tagger (POS) from which nouns, noun phrases, and verbs can be identified. Next module is OWL parser which parses the ontology represented in OWL and thus the classes, ObjectProperties and DataProperties are extracted. Now the nouns and noun phrases become the candidate for entities and attributes identification. In the last two steps domain ontology is used to explore the important concepts of the domain. Identification of entities and their attributes is followed by extraction of key attributes for the entities. Thus, at the end we get the desired relational database schema.

Figure -2: Architecture of the proposed system

Following section describes each module in detail:

POS tagger: POS tagger i.e. Part Of Speech tager will parse the entire text document of user requirements and will tag each word according to its part of speech (e.g. Noun, verb, adjective, adverb, etc.) and extract all the verbs and nouns from it. Now nouns serve as a source of attribute and class identification.

OWL Parser: It parses the extracted nouns and verbs using OWL parser. In OWL i.e. Ontology Web Language, domain specific conceptual terms are termed as entities, relationships are termed as ObjectProperty and attributes are represented as DataProperty. So, with the help of OWL Parser these are extracted and stored.

Entities and Attributes extractor: It extracts the components like entities and attributes from the nouns and noun-phrases. Domain ontology is used here to extract contextual entities and attributes. Attributes are specified as DataProperty in OWL. If domain ontology does not contain any information about the nouns and noun phrases under consideration, then the tagged nouns and noun phrases are again processed and semantic similarity is taken into consideration in such cases.

Key attribute Extractor: A rule-based approach is used to identify the primary key attribute from the attributes of all
the tables. Following this, another rule based approach helps in retrieving foreign keys and their associated entities.

4. IMPLEMENTATION

The system is implemented using Python Language. The input for the system is Problem statement and Problem domain. The problem statement is in the form of natural text which is then Tokenized, Tagged and Lemmatized using spaCy[8]. spaCy is an open-source software library for advanced Natural Language Processing, written in the programming languages Python and Cython. Based on the input problem domain a suitable Domain Ontology is webscraped and parsed using libraries such as BeautifulSoup and OntoSpy. Nouns from the tagged problem statement are classified as Entities and attributes based on lexical and semantic similarity with those extracted after parsing the OWL file that was webscraped based on suitable input Domain ontology. To tackle the ambiguity caused due to use of Natural Language in problem statement and to overcome lexical dissimilarity of common contextual words, the system makes use of Google Dictionary to locate probable synonyms in the problem statement. The schema obtained is then marked with Primary keys and Foreign keys according to the relationship between the entities with rule-based algorithms that is explained in detail in the following paragraph. The system aims to provide user flexibility and thus recommends them few extra appropriate and probable entities and attributes that are fit for the problem at hand and can be included in the Schema on need basis. An extra functionality provided by our tool is retrieval of relational database schema if the user wishes to only submit the domain name or theme of the project and not the entire customized problem statement. In this scenario, we simply parse the OWL file of the respective Domain Ontology and provide with the relevant classes, attributes, key attributes and recommended schema related elements.

Rule-based Algorithm for determining Primary keys: -

FOR EACH attribute IN attribute list of an entity

1. Find attributes with substring "_no_/number/_ID" and String_1=Split and extract the word lying before "_no_/number/_ID"

2. IF String_1 matches with Entity name, THEN it qualifies to be the primary key.

3. ELSE IF Any of the Meanings/Synonyms (that are retrieved through Google dictionary) of String_1 matches with the entity name, THEN it qualifies to be the primary key.

4. ELSE no primary key exists and the entity qualifies to be a weak entity.

Rule-based Algorithm for determining Foreign keys: -

FOR EACH attribute IN attribute list of an entity

1. Find attributes with substring "_no_/number/_ID" and String_1=Split and extract the word lying before "_no_/number/_ID"

2. IF String_1 does not match with the Entity name(E) and matches with one of the Entity name in the Entity list(L) THEN it qualifies to be the foreign key of E with the parent Entity being the matched Entity name from L.

3. ELSE IF Any of the Meanings/Synonyms (that are retrieved through Google dictionary) of String_1 does not match with the Entity name(E) and matches with one of the Entity name in the Entity list(L) THEN it qualifies to be the foreign key of E with the parent Entity being the matched Entity name from L.

4. ELSE IF String_1 does not match with the Entity name(E) and matches with the Synonyms of one of the Entity names in the Entity list(L) THEN it qualifies to be the foreign key of E with the parent Entity being the matched Entity name from L.

5. ELSE no foreign key exists.

Let us consider an example of an "Hospital Management System". The problem domain entered by the user is "Hospital" and the problem statement entered by the user which is in the natural language based textual format is as follows:

"Patients request for appointment for any doctor by specifying the doctor_name and doctor_id. The details of the existing patients like Name, Address, age and gender are retrieved by the system. New patients update their details in the system using a unique patient_id allotted to them before they request for appointment with the help of assistant_name where each assistant can be distinguished based on their distinct Assistant_id. The assistant confirms the appointment based on the availability of free slots for the respective Medical practitioners and the patient is informed. Assistant may cancel the appointment at any time."

Firstly, the input text is tokenized, tagged and lemmatized. Secondly, "hospital.owl" i.e. an Ontology based on user entered problem domain is retrieved using web scraping. Thereby, the nouns such as Patients, Doctor, Assistant, slots, etc. are classified as entities and Name, Address, age and gender, doctor_name, etc. are classified as attributes with the help of the ontology. Ambiguous words like "Doctors" and "Medical Practitioners" are handled further to avoid redundant entities and attributes. The primary key and foreign key of each entity are identified using appropriate rule-based approach. Thus, a database schema is generated.
The expected output is as follows:

<table>
<thead>
<tr>
<th>Patients</th>
<th>Doctor</th>
<th>Assistant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Doctor_name</td>
<td>Assistant_name</td>
</tr>
<tr>
<td>Gender</td>
<td>Specialization</td>
<td>Degree</td>
</tr>
<tr>
<td>Age</td>
<td>Degree</td>
<td>Address</td>
</tr>
<tr>
<td>Address</td>
<td>Address</td>
<td>Age</td>
</tr>
<tr>
<td>Patient_id</td>
<td>Doctor_id</td>
<td>Assistant_id</td>
</tr>
</tbody>
</table>

The system generates a schema for the given problem statement and identifies Primary keys and Foreign keys.

5. CONCLUSION

The Automatic Database Schema Generator tool has been successfully implemented for extracting database schema from natural language textual input problem statement. The tool can analyze words with similar context and integrates the results accordingly. The keys have been identified to denote referential integrity and uniquely identify the entities. An entire repository of commonly used Ontologies proved to be favorable for mapping various attributes to their respective entities. Thus, a well enriched database schema is retrieved with the help of this tool.

REFERENCES


[4] Class diagram extraction from textual requirements using Natural language processing (NLP) techniques Mohd Ibrahim Al-Qaoud, Rodina Ahmad Department of Software Engineering, Faculty of Computer Science and Information technology, University of Malaya, Malaysia


[8] spaCy:https://spacy.io/Websites