Natural Language Querying in SAP-ERP Platform

Diptikalyan Saha
IBM Research, India
diptsaha@in.ibm.com

Neelamadhav Gantayat
IBM Research, India
neelamadhav@in.ibm.com

Senthil Mani
IBM Research, India
sentmani@in.ibm.com

Barry Mitchell
IBM Global Business Services, USA
bcm@us.ibm.com

ABSTRACT

With the omnipresence of mobile devices coupled with recent advances in automatic speech recognition capabilities, there has been a growing demand for natural language query (NLQ) interface to retrieve information from the knowledge bases. Business users particularly find this useful as NLQ interface enables them to ask questions without the knowledge of the query language or the data schema. In this paper, we apply an existing research technology called "ATHENA: An Ontology-Driven System for Natural Language Querying over Relational Data Stores" in the industry domain of SAP-ERP systems. The goal is to enable users to query SAP-ERP data using natural language. We present the challenges and their solutions of such a technology transfer. We present the effectiveness of the natural language query interface on a set of questions given by a set of SAP practitioners.

CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**;
• **Information systems** → *Ontologies*;

KEYWORDS

Natural Language Query, Natural Language Interface, SAP

ACM Reference format:
Diptikalyan Saha, Neelamadhav Gantayat, Senthil Mani, and Barry Mitchell. 2017. Natural Language Querying in SAP-ERP Platform. In *Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17),* 6 pages.
https://doi.org/10.1145/3106237.3117765

1 INTRODUCTION

The omnipresence of mobile devices coupled with recent advances in automatic speech recognition capabilities resulted in a growing demand for natural language interfaces to perform information retrieval or perform data processing. For example, we have seen natural language interface for program synthesis [8], programming by example [14], etc.

In this work, our aim is to build a Natural Language Query Interface to the SAP-ERP system. Most business users (or customers) use custom applications, or SAP-GUI to run transactions (programs) to interact with the SAP's underlying data-store. Typical business users must undergo various training programs to gain sufficient expertise to interact with the SAP system or rely on experts to answer their queries. Due to vast business process coverage of SAP-ERP, we often see that even the experts have knowledge of only one of the many SAP functional areas. Therefore, the ease of access to entire SAP-ERP data by the common business users is far from ideal. Our goal is to make the information querying and extraction easy for business users by providing a natural language query interface. Business users can now essentially ask questions for any domain through our system without the technical knowledge of how to reach to a data through the maze of multiple programs and options.

ATHENA [17] is an Ontology-driven natural language query engine for relational database. ATHENA enables database access to naive users who do not know database query languages such as SQL. Even for those who do possess the technical proficiency, natural language interface obviates the need to know the exact schema of the database; it is simply sufficient to know the semantic information and its scope captured in the database.

As SAP-ERP system internally uses relational databases to store information, ATHENA can therefore serve as a framework for natural language interface for SAP-ERP data. Moreover, the core of ATHENA is independent of any domain and can be instantiated for different domains using domain ontologies making it a suitable framework for SAP-ERP instantiation. To the best of our knowledge, this is the first attempt to develop natural language interface to access SAP-ERP data.

Instantiating ATHENA framework for SAP-ERP domain posed many challenges. Creating an Ontology/domain schema consisting of concepts, properties, and relationships for the SAP domain is a daunting task. Therefore, we developed an automatic method for inferring domain schema from the database schema. Apart from the Ontology, ATHENA also requires domain vocabulary which consists of 1) a translation index which maps data value (and its variations) in the tables to the Ontology and 2) synonyms associated with the various Ontology elements. Once the Ontology and associated domain vocabulary is setup, ATHENA can interpret natural language questions. However, this results into a performance problem due to the size of the inferred Ontology. We used an Ontology partitioning technique to overcome the performance problem.

The contributions of the paper are listed below:

- We automatically create an Ontology capturing domain schema for SAP-ERP system.

ESEC/FSE'17, September 4–8, 2017, Paderborn, Germany

ACM ISBN 978-1-4503-5105-8/17/09…$15.00
https://doi.org/10.1145/3106237.3117765

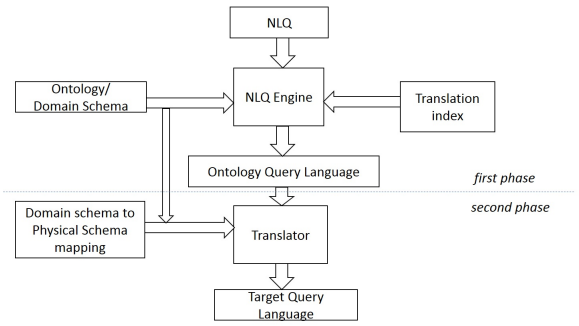

Figure 1: High Level Architecture of ATHENA

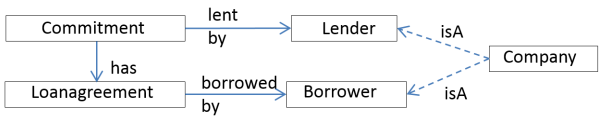

Figure 2: A Semantic graph of a part of the Finance Ontology.

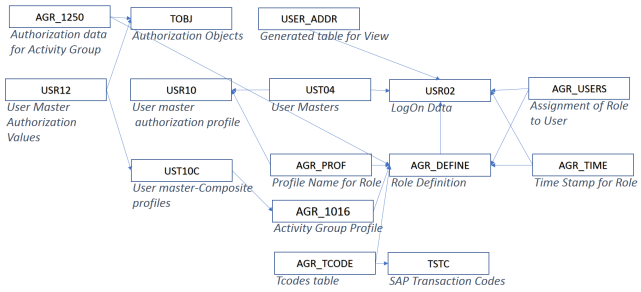

Figure 3: A part of SAP Ontology for Security Administration

- We successfully create a usable domain vocabulary (synonyms associated with Ontology elements and variation of data in the Ontology) for instantiating the ATHENA.
- We use an Ontology partitioning procedure to overcome the performance problem in ATHENA.
- We show the effectiveness of our technique for user given questions in SAP domain.

The rest of the paper is organized as follows. In Section 2, we present the background on ATHENA work. In Section 3, we show how we instantiate ATHENA system for NLQ domain. We demonstrate the effectiveness of our system in Section 4. We present the related work in Section 5 and conclude with our learnings in Section 6.

2 BACKGROUND

In this section, we present some background knowledge on ATHENA.

The high-level architecture of ATHENA is presented in the Figure 1. As evident from the figure, ATHENA follows a two-phase approach. In the first phase, the natural language question is translated to an intermediate language called Ontology Query Language (OQL). In the second phase, the OQL query is translated into an SQL query. Having this two-phase approach offers physical to logical independence as OQL query is not dependent on the physical schema. Moreover, it is possible to instantiate ATHENA for different types of data stores (e.g. RDF stores, Titan Graph store, Solr/Document store) by plugging in different OQL to data stores specific translators.

ATHENA takes an Ontology as input which captures the domain schema. It contains classes or concepts, their properties and the set of relationships between concepts. Together they are called Ontology elements. In the context of ATHENA, the Ontology does not contain any data i.e. instances of Ontology elements, even though the generic notion of Ontology can contain instance data. The input natural language query is interpreted over the Ontology. The advantage of using Ontology is two-fold. First, ATHENA can be instantiated for different domains by constructing different domain ontologies. Secondly, it presents richer information than just database table schema and dependency between tables and therefore helps in creating precise interpretation.

ATHENA requires three inputs: Ontology, domain vocabulary and Ontology to database mapping for domain specific instantiation.

1. Ontology. We use the notion of Ontology to represent the schema of the domain in terms of classes/concepts (C), and their properties (P) and the set of relationships (or relations) as $R (\subseteq C \times C)$, collectively called as Ontology elements. The relations can be of two types: *inheritance* and *functional*. A semantic graph $G = (V, E)$ is an directed graph constructed from an Ontology where $V = C$ and $E = R$ i.e. edges exist between concepts taking part in a relation. One such graph is shown in Figure 2 taken from [17]. Note that many different subclasses of the concept `Company` are present in the Ontology (e.g. `Lender`, `Borrower`, etc.), each performing a different role in the domain. The semantics of edge direction is determined by the subclass/superclass role and the arity of functional relations.

2. Domain Vocabulary. Each Ontology element has a set of synonyms associated with it. The synonyms are required to map the tokens in the NLQ to the Ontology elements. For example, a `Company` concept can have synonyms like `Institution, Business, Corporation, Organization`.

Another index (called *Translation Index*) is maintained which maps the data values and their variations to the corresponding Ontology properties. For example, two sample entries in the index can be '*DeltaAirlines*' $\mapsto$ *company.name*, *Delta Airlines* and *Delta* $\mapsto$ *company.name*, *Delta Airlines* where 'Delta' is a variation for 'Delta Airlines'. Note that multiple companies like 'Delta Security' and 'Delta Airlines' can have the same variation. The translation index is generated offline and updated with the database changes.

3. Ontology to Database Mapping. This mapping essentially bridges the gap between logical and physical schema. When relational back-end is used, an obvious mapping can be used between concept to tables, properties to fields, and relationship to foreign-key constraint.

We explain in detail how these inputs are generated in the context of SAP-ERP domain in the following section. Below we present

a short account of how ATHENA transforms an NLQ to SQL.

NLQ Engine. We explain the procedure with the help of an example from the SAP Security domain - "Show me list of users with SAP_ALL access" against the Ontology shown in Figure 3. In the first step, the tokens of the NLQ are mapped into Ontology elements. This mapping is called *evidence*. In this case, the token `user` gets mapped to multiple Ontology elements (called *candidates*) like `USR02.BNAME, AGR_USERS.BNAME, UST04.BNAME` using synonyms (called meta-data evidence). The token `SAP_ALL` gets mapped to (called data-value evidence) `UST04.PROFILE` using the Translation Index.

In the second step, a set of *selected set*s are computed where each *selected set* is formed by taking one candidate for each matched token. In this case, 3 selected sets are formed. For each selected set, ATHENA tries to build an interpretation where an interpretation is represented as a subgraph in the semantic graph connecting the elements in the selected set. Doing so, semantically grounds the words in natural language to specific meanings by referring to elements in the semantic graph. Note that, many interpretations can arise since there can be many possible subgraphs connecting elements of a selected set in addition to there being many possible selected sets. ATHENA uses Steiner Tree based algorithms [10] to generate a single interpretation of minimal size for a *selected set*. Further it employs a goodness criterion to choose one or more interpretations across all *selected sets*. In this example, in one of the *selected set*s, both the tokens got mapped to the same concept (`UST04`) which results in the smallest interpretation tree containing only one node across all the *selected set*s.

In the third step, an OQL is generated from an interpretation tree and an annotation procedure. OQL is like SQL in terms of syntax but refers to the Ontology elements instead of database schema. For example, using the annotation procedure, `users` is identified as the return entity and associated to the select clause. The only concept (`UST04`) goes to the FROM clause. The index value match results into an equality predicate in the WHERE clause. The resultant intermediate OQL query is generated is `SELECT oUST04.BNAME FROM UST04 oUST04 where oUST04.PROFILE = 'SAP_ALL'`. The OQL to SQL translation is performed using the Ontology and database mapping.

3 SAP-ERP INSTANTIATION OF ATHENA

In this section we discuss in detail the various challenges and our approach in instantiating the ATHENA solution for SAP-ERP domain.

3.1 SAP Schema Ontology

The first step infers the SAP Ontology from SAP-ERP database schema. The mapping between Ontology and database tables is as follows. Each concept in the Ontology corresponds to a database table. The properties correspond to the table fields which are not the foreign keys. The relationships between two concepts are obtained from the primary key-foreign key relationship.

Apart from this straightforward mapping, we need to infer the type of relationship (functional/inheritance) and cardinality of the functional relationship. According to SAP experts, SAP does not keep Foreign Key - Primary Key relationship for inheritance. In case there are multiple subclasses of a superclass, SAP has a single table where a field in the table signifies the type of the subtype in the table. Consider the example of `VBAK` table which stores SAP Sales Document: Header Data. It has a field called `VBTYP` which signifies different types of documents like Inquiry, Quotation, Order, Item proposal, etc. Typically this could have been expressed by a table called *Sales Order document* and individual subtype tables for different types of documents. Following SAP's table structure, we do not infer any inheritance relationship.

Table 1: Ontology information source from Data Dictionaries

Ontology	SAP Database Information	Example
Concept	DD02L/TABNAME	MBEW
Concept/Property Map	DD03L/TABNAME	MBEW
Property (Name)	DD03L/FIELDNAME	MATNR
Property (Type)	DD03L/DATATYPE, DD03L/LENG	CHAR, 00040
Relationship (CONCEPT1)	DD08L/TABNAME	MBEW
Relationship (CONCEPT2)	DD08L/CHECKTABLE	MARA
Relationship (Name)	DD08L/FIELDNAME	MATNR
Relationship (CARDINALITY)	DD08L/CARDLEFT, DD08L/CARD	CN, 1

Table 2: Example of synonym source.

DD02T

TABNAME	TTEXT
MARA	General Material Data
MBEW	Material Valuation

DD04T

ROLLNAME	DDTEXT
MATNR	Material Number
LBKUM	Total Valuated Stock

SAP additionally maintains a set of data dictionary tables which contain information regarding the tables in SAP. The Table 1 lists the required data dictionary tables along with example values from which we can get necessary information to create an Ontology. The example shows a relationship between two tables (`MBEW` and `MARA`) from the domain Material Management(MM).

3.2 Synonyms

Synonyms need to be associated with Ontology elements such that tokens in the NLQ can be mapped to the Ontology elements.

In prior efforts, ATHENA used hand created ontologies. Therefore, Ontology elements had meaningful names. Synonyms of such names could be found using online dictionary lookup. For example, `Company` was a concept in the finance Ontology and the synonyms for `Company` can be easily extracted by lookup in thesaurus [6].

The challenge with SAP is that all tables and field names are non-intuitive (e.g. MARA, VBAK). Fortunately, SAP's data dictionary tables again come to the rescue. The description of the table and field names are available at *DD02T* and *DD04T* tables respectively. An example is provided in Table 2.

Now the challenge is to create synonyms from these descriptions. Earlier, ATHENA has been tried for single word names of the Ontology elements. Synonyms of a single word can be easily obtained using a thesaurus. However, in SAP most field descriptions have multiple words/phrase. This creates an additional challenge for creating appropriate synonyms of the description. For example, `Role Name` is the description corresponding to a field name `AGR_NAME` of the table `AGR_DEFINE`. Now `role` and `name` should be synonyms of `Role Name`. Therefore, we also perform POS tagging for noun detection to get constituent words as synonyms. For multiple words,

we get the nouns and the verbs and replace those words in the description with their synonyms to generate variations.

3.3 Translation Index

The Translation Index (TI) captures the mapping between the data values and their variations to the Ontology elements. More precisely, each entry in the translation index contains the mapping between a variation (which could be same as original value) of the data value in a table field to the pair consisting of the original data value and the Ontology property corresponding to the table field. The variation helps in mapping data values mentioned in NLQ to the Ontology elements and the actual value is required to create the appropriate predicates.

There are three challenges in the creation of TI - 1) which properties to select for creating the Translation Index, 2) how to index ENUM fields, and 3) how to create value variations.

Earlier ATHENA was used for ontologies having a maximum size of 100 concepts. Therefore, manual identification of the properties for creating Translation Index was possible. For SAP, we initiated with all key properties in Master table and all ENUM properties (which has fixed number of values). Such heuristic worked for our corpus of frequently asked user questions.

There are some table fields (called ENUM fields) which have a fixed number of values. Each value is a character or two. For example, `VBAK.VBTYP` has 69 possible entries, e.g. L, M, N etc. The corresponding meanings of the values are Debit memo request, Invoice, and Invoice cancellation, respectively. An NLQ will refer to the meanings of these ENUM values. Therefore, in Translation Index, the variation of value "L" should contain its description "Debit Memo". SAP's data dictionary table `DD07T` keeps the mapping between field's domain (`DD07T.DOMNAME`) and corresponding values (in `DD07T.DDTEXT`). The information of a field's domain is obtained from `DD03L.DOMNAME`.

3.4 Ontology to Physical Schema Mapping

The physical-logical bridge is described using this mapping. As in this case, the Ontology was created from the database schema, such mapping is straightforward, as described in Section 2.

3.5 Query Translator and Execution

ATHENA employs a translator to convert OQL query to an SQL query. In the SAP domain, we encountered few issues regarding query translation and execution.

SAP system uses relational databases such as Oracle, HANA, DB2 and MS SQL Server as their back-end database. Each DBMS supports some variety of SQL syntax (called Native SQL). To overcome the issue of interoperability, SAP created its own language called *Open SQL*. SAP internally translates the Open SQL syntax to Native SQL syntax based on its underlying database at runtime. SAP also gives the provision of using Native SQL syntax in ABAP programs. Therefore we actually have two choices for translation - Native and Open SQL. The Open SQL interface is usually preferred over native access because it is independent of the underlying database which makes it more tolerant of data model evolution and customization.

Dynamic query execution is an even more complicated issue in SAP. It is possible to completely bypass SAP system and execute the query into the underlying database. It is therefore not a recommended way. Another tedious way is to generate an ABAP program on-the-fly (program synthesis) and register the program by means of SAP transport and then execute it. This approach has its drawbacks in performance, maintenance, and monitoring. We found an article [1] which describes how to perform dynamic Open SQL and implemented this methodology to realize the query translation and execution module.

In ATHENA, the translator component produces SQL (SQL-92 compatible) from OQL. We have made changes to the translator to produce separate clauses of Open SQL [2]. The Query execution takes these clauses and connects to the SAP-ABAP system using the JCo Connector [4] and executes the ABAP Function module responsible for executing the dynamic Open SQL.

3.6 Performance Optimization based on Ontology Partitioning

One of the inherent problems of ATHENA's algorithm is that it generates a set of *selected sets*, each of which is formed by choosing one candidate from the evidence set of each token. The number of *selected sets* can be potentially very large if one or more tokens have many candidates. This resulted in performance degradation (response time more than one second). For smaller ontologies, the performance degradation may not be serious. However, for large ontologies, this is a serious issue. Such case is commonly seen for NLQ with date/time expressions. ATHENA uses an in-house developed TIMEX grammar to parse date/time/duration related expression. Then it maps the recognized token to all properties having DATE or TIMESTAMP type. Typically, there are multiple such date related properties in an Ontology - which can result in a large number of candidates.

In this paper, we present a solution to address this problem. We partition the Ontology based on SAP modules/sub-modules. Each component of the partition contains a set of concepts (and their properties). However, the components may not be disjoint as some SAP master tables are used in multiple domains. The set of *selected sets* are pruned if the *minimum number of components* of the tokens exceeds one. Note that, as each token can be mapped to multiple components due to overlapping nature of the partition, a *selected set* is associated to different combination of components. Therefore, it is required to compute the minimum number of components. The only exception to this rule is for a unique candidate of a token. If a token is mapped to a unique Ontology element, then the candidate needs to be there in all *selected sets*.

The intuition of this optimization comes from the fact that each individual question asked by a practitioner pertains to a single domain in SAP. Some of the example domains and their codes are - Sales and Distribution (SD), Material Management (MM), Financial Accounting - Controlling (FI/CO), Accounting (AC). One can view such classification using shortcuts from SE11 Transaction [3]. Each domain is divided into several sub-domains. We map each table into a sub-domain using the following query.

```
SELECT TADIR.OBJECT, DF14L.PS_POSID, DF14L.name
FROM TADIR INNER JOIN  TDEVC ON TADIR.DEVCLASS = TDEVC.DEVCLASS
INNER JOIN DF14L ON TDEVC.COMPONENT = DF14L.FCTR_ID
WHERE TADIR.PGMID = 'R3TR' AND TADIR.OBJECT = 'TABL';
```

Table 3: (A) Ontology, (B) Partition Characteristic

Characteristics	Size
Concepts	513,985
Properties	9,692,032
Relations	897,031

(A)

Sub-domain	Concepts
Contract Act. Receivable & Payable	7898
Basic Functions	6499
Claims Management	5323
European Monetary Union: Euro	5240
Master Data	4395
Production & Revenue Accounting	3770
Obsolete Functionality	3557
Incentive & Commission Mgmt.	3520
Administration	3425
Advertising Management	3277

(B)

Based on the above query both MARA and MBEW belong to the same domain (LO-MD-MM with description Material Master). The mapping between the sub-domains to domains is obtained using SE11 Transaction's menu called SAP Application [3].

Partitioning can be helpful for 1) pruning out some parts of the Ontology, 2) visualizing large ontologies, and 3) creating NLQ prototype for a particular SAP domain, especially for testing.

4 EXPERIMENTAL RESULTS

In this section, we present an empirical account of the instantiation of the ATHENA system for SAP-ERP domain. We present the experimental results in three categories - 1) Ontology generation, 2) experiments performed with user questions, and 3) Ontology partitioning. For all experiments, we have used an SAP HANA system. All our code is written in Java and experiments are conducted in Thinkpad running Windows 7 with 2.54 Ghz processor, 8GB RAM running Java 8. We used 128GB RAM machine to load the translation index.

Ontology Generation and Partitioning. Using the method described in Section 3, we created the Ontology. The characteristics of the Ontology/domain schema is shown in Table 3(A). Note that, as we do not infer inheritance relation, all relations are deemed functional.

Next, we partition the Ontology based on the SAP domains. In Table 3 we present the number of concepts for first few large sub-domain, as derived by the SQL query given in the last section. We could generate 2261 sub-domains which are further divided into 63 domains shown by SAP [3]. Although the average number of concepts for each sub-domain is 227, the distribution of the concepts across sub-domains is skewed (as shown in Table 3).

Handling NLQs. We obtained a list of questions from SAP functional experts in three modules. They were asked to give the most relevant question which are subsequently run against SAP-ATHENA.

Out of 42 questions, ATHENA was not able to answer 1 question (did not return any answer), for others, it returned the correct answer (precision 100%, recall 97%), manually verified by the practitioners. We discuss the failure case here.

Consider the Question #41. In this case, the phrase `create purchase order` and `clear invoices` get mapped to `TSTC.TCODE`. In this case ATHENA fails to identify the semantics of why two tokens are mapped to the same phrase. Specifically, it cannot identify whether the query can be 'union' case or an 'intersection' case. Handling this case is taken as a future work in ATHENA.

Performance. All queries in ATHENA took less than 1 second. We now present the potential performance problem with question #42. If partitioning is not used, the time expression *last 30 days* is mapped to all date-time properties in the Ontology. The count of date-time properties in SAP Ontology is 509K which makes this a real problem (ATHENA did not terminate within 1 minute). In SAP administration domain, the count is only 30 which makes it tractable. Even though the average number of date-time properties per sub-domain is 225, the maximum reaches to 10292 for 'Contract Accounts Receivable and Payable'.

Table 4: Questions gathered from SAP practitioners in the field of Sales Distribution, FI/CO, Security Administration modules

#	User Question
1	List of customer for whom sales is blocked
2	List of customers marked for deletion
3	List of third party vendors that are pending for delivery
4	How many down payments are pending from the customers
5	How many orders partially delivered
6	Show me the orders where partially billing happened
7	How many orders fully delivered
8	How many products are rejected
9	List of customers associated with Germany
10	List of materials associated with Germany
11	Find all materials which can be delivered to Germany
12	What all customers to whom we can deliver in Germany
13	Which are all the valid contracts still pending
14	Quantity per contract
15	How many GL accounts extend to company codes
16	How many GL accounts are blocked for posting
17	How many GL accounts are marked for deletion
18	How many customers are blocked for posting
19	How many customers are blocked for payment
20	How many customers are marked for deletion
21	How many vendors are blocked for posting
22	How many vendors are blocked for payment
23	How many vendors are marked for deletion
24	Materials which should not be used for costing
25	For how many customers logistic payments are blocked manually
26	For how many customers logistic payments are blocked due to existence of blocking reason
27	For how many customers logistic payments are stochastically blocked
28	List vendors with Resident G/L account
29	List Assets with Resident G/L account
30	List Materials with Resident G/L account
31	List G/L accounts with Resident G/L account
32	List Customers with Resident G/L account
33	Show me a list of users with SAP_ALL access
34	Show list of users having DEBUG access in production system
35	Show list of users having assess to import transport requests
36	Show list of users having access to OB52 tcode for opening and closing the periods
37	Show list of users whose password is not changed after ID creation
38	Show list of ROLES having object status as MANUAL in the role
39	Show me all Job Roles that have access to transaction SA01
40	How many users have access to transaction SA01
41	Do any users have access to create purchase orders and also clear invoices
42	Show me a list of users in the UK who have not logged on in the last 30 days

Threats to Validity. We have gathered a small set of questions from the users to demonstrate the feasibility of our solution in ERP domain. In future, we intend to have a large-scale experimentation. Also, the information in data-dictionary tables varies from SAP system to system. Therefore, we need to study the difference between systems and its impact on automatic extraction of the Ontology.

We have noticed that there are some relationships missing (as per entries in DD08L) from the Ontology. For example, there are two tables USR01 and USR02 in the domain of SAP administration which has same key property BNAME (user name) and set of values, which are not explicitly mentioned in DD08L. We believe that such missing relationships can be inferred from other references to these tables in the data dictionary, and will enable ATHENA to construct queries involving both the tables.

5 RELATED WORK

We discuss related work in two categories - NLQ applications and NLQ technology.

NLQ Applications. NLQ over relational database has been applied in various benchmarks (data and domains) such as Finance [17], GEO [12, 17], Microsoft Academic Search [12], Restaurant [18]. NLQ has been translated to PL/SQL for general purpose Oracle database in [9]. However, to the best of our knowledge, natural language querying is not addressed for any ERP domain, and our work is the first in applying this to SAP-ERP.

NLQ Technology. There have been works to interpret the semantics of a full-blown English language query. These works, in general, try to disambiguate among the potentially multiple meanings of the words and their relationships. Some of these are machine learning based [7, 15, 18] that require good training sets, which are hard to obtain. Others require user feedback [11–13]. However, an excessive user interaction to resolve ambiguities can be detrimental to user acceptance.

Most of the non-learning based disambiguation techniques (e.g., [16]) build on database integrity constraints and thus do not capture the rich semantics available in the Ontology. In SAP-ATHENA, we also could not exploit the rich information (inheritance and additional relationship types). We still differentiate with them in two key aspects - none of these techniques used partitioning to cope with the large databases. Existing database techniques translate NLQ to SQL. But, our two-stage framework was amenable for translation to other target languages such as Open SQL.

6 CONCLUSION

Impact. SAP has approximately 12 million users worldwide [5]. A natural language interface to SAP's underlying data can potentially benefit all customers who want to extract information from SAP's knowledge base.

Summary. In this paper, we described how we have used an existing natural language query framework called ATHENA for an industry use case for SAP-ERP systems. ATHENA is Ontology/domain schema driven and therefore we have presented a way to automatically extract domain schema from SAP table schema. Because of the huge domain schema and vocabulary, most of our solution had to be automated which was not the case in ATHENA's previous applications. We relied heavily on SAP's data dictionary tables to provide all information which we could use directly (Ontology creation) or further processed (synonym or variant creation) to provide all information required to instantiate ATHENA framework for SAP. Apart from engineering challenges like creating a translator from ATHENA intermediate query language (OQL) to Open SQL, the scale of domain schema motivated us to extend ATHENA's core solution. We used Ontology partitioning, automatically derived from SAP's domain to table mapping, to address performance bottleneck of some particular classes of the queries.

Limitations and Future Work.

- During the testing phase, we encountered that ATHENA's limitations in handling nested query.
- ATHENA can translate an NLQ into a SQL query. However, some user intent may not be expressible using a single query (may require generating ABAP programs).
- We plan to analyze SAP tables to infer inheritance and union relationships and missing functional relationships.
- We want to try our systems with more user questions. IBM has around 50K SAP practitioners. Currently, the system is used by only a few of them. We plan to first release this system to IBM practitioners and collect feedback.
- We believe that we can use a sub-domain based ranking mechanism to further prune out interpretations.
- The system could be extended for making insert/update to the tables, to support live transactions.

It was surprising to learn from SAP experts that most users actually ask simple questions which do not require join for more than 4 tables. We therefore hope that our system will be useful to millions of user worldwide.

ACKNOWLEDGMENTS

We thank all the IBM Global Business service experts including Anjan Nandy, Manas K. Das, Ashish Goutam, Sandeep Singh, Ragini Thothathiri, Charudatta S Pande who provided us the set of questions and their expert SAP knowledge.

REFERENCES

[1] Dynamic Open SQL. (Dynamic Open SQL). Retrieved July 20, 2017 from https://www.ibm.com/developerworks/data/library/techarticle/dm-1007sapopensql/
[2] Open SQL. (Open SQL). Retrieved July 20, 2017 from https://help.sap.com/saphelp_nw70ehp1/helpdata/en/fc/eb3983358411d1829f0000e829fbfe/content.htm
[3] SAP - module wise tables. (SAP - module wise tables). Retrieved July 20, 2017 from SAPGUI,SE11,PRESSF4,SAPAPPLICATIONSMENU
[4] SAP JCo Connector. (SAP JCo Connector). Retrieved July 20, 2017 from https://help.sap.com/saphelp_nwpi711/helpdata/en/48/70792c872c1b5ae10000000a42189c/content.htm
[5] SAP User Statistics. (SAP User Statistics). Retrieved July 20, 2017 from https://goo.gl/MeFwVa
[6] THESAURUS. (THESAURUS). Retrieved July 20, 2017 from https://www.merriam-webster.com/thesaurus/company
[7] Jonathan Berant et al. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*. 1533–1544.
[8] Aditya Desai, Sumit Gulwani, Vineet Hingorani, Nidhi Jain, Amey Karkare, Mark Marron, Sailesh R, and Subhajit Roy. 2016. Program Synthesis Using Natural Language. In *Proceedings of the ICSE (ICSE '16)*. ACM, New York, NY, USA, 345–356. https://doi.org/10.1145/2884781.2884786
[9] Swapnil Kanhe, Pramod Bodke, Akshay Chikhale, and Vaibhav Udawant. 2015. SQL Generation and PL/SQL Execution from Natural Language Processing. In *International Journal of Engineering Research and Technology*, Vol. 4. IJERT.
[10] L. Kou, G. Markowsky, and L. Berman. 1981. A Fast Algorithm for Steiner Trees. *Acta Informatica* 15, 2 (1981), 141–145.
[11] D. Küpper, M. Storbel, and D. Rösner. 1993. NAUDA: A Cooperative Natural Language Interface to Relational Databases. In *ACM SIGMOD*.
[12] Fei Li and H. V. Jagadish. 2014. Constructing an Interactive Natural Language Interface for Relational Databases. *Proc. VLDB Endow.* 8, 1 (2014), 73–84.
[13] Yunyao Li, Huahai Yang, and H. V. Jagadish. 2005. NaLIX: An Interactive Natural Language Interface for Querying XML. In *ACM SIGMOD*.
[14] Mehdi Manshadi, Daniel Gildea, and James Allen. 2013. Integrating Programming by Example and Natural Language Programming. In *AAAI (AAAI'13)*. AAAI Press, 661–667. http://dl.acm.org/citation.cfm?id=2891460.2891552
[15] Ana-Maria Popescu et al. 2004. Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability. In *COLING*.
[16] Etzioni Oren Popescu, Ana-Maria and Henry Kautz. 2003. Towards a Theory of Natural Language Interfaces to Databases. In *IUI*.
[17] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R. Mittal, and Fatma Özcan. 2016. ATHENA: An Ontology-driven System for Natural Language Querying over Relational Data Stores. *Proc. VLDB Endow.* 9, 12 (Aug. 2016), 1209–1220.
[18] Lappoon R Tang and Raymond J Mooney. 2001. Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. In *ECML*.