Conversational OLAP

(Discussion Paper)

Matteo Francia[1], Enrico Gallinucci[1] and Matteo Golfarelli[1]

[1]DISI - University of Bologna, Italy

Abstract

The democratization of data access and the adoption of OLAP in scenarios requiring hand-free interfaces push towards the creation of smart OLAP interfaces. In this paper, we describe COOL, a framework devised for COnversational OLap applications. COOL interprets and translates a natural language dialog into an OLAP session that starts with a GPSJ (Generalized Projection, Selection, and Join) query and continues with the application of OLAP operators. The interpretation relies on a formal grammar and on a repository storing metadata and values from a multidimensional cube. In case of ambiguous text description, COOL can obtain the correct query either through automatic inference or user interactions to disambiguate the text.

Keywords

Natural language processing, OLAP

1. Introduction

Following the spreading of analytic tools, a heterogeneous plethora of data scientists is accessing data. However, the gap between data scientists and analytic skills is growing since different types of data require to learn specialized metaphors and formal tools (e.g., SQL language to query relational data). Natural language interfaces are a promising bridge towards the democratization of data access [1]. Rather than demanding vertical skills in computer science and data architectures, natural language is a native "tool" to organize and provide meaningful questions/answers. Interfacing natural language processing (either written or spoken) to database systems opens to new opportunities for data exploration and querying [2].

Actually, in the area of data warehouse, OLAP (On-Line Analytical Processing) is an *"ante litteram"* smart interface, since it supports the users with a "point-and-click" metaphor to avoid writing well-formed SQL queries. Nonetheless, the possibility of having a conversation with a smart assistant to run an OLAP session (i.e., a set of related OLAP queries) opens to new scenarios and applications. It is not just a matter of further reducing the complexity of posing a query: a conversational OLAP system must also provide feedback to refine and correct wrong queries, and it must have memory to relate subsequent requests. A reference application scenario is augmented business intelligence [3], where hand-free interfaces are mandatory.

SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy

m.francia@unibo.it (M. Francia); enrico.gallinucci@unibo.it (E. Gallinucci); matteo.golfarelli@unibo.it (M. Golfarelli)

0000-0002-0805-1051 (M. Francia); 0000-0002-0931-4255 (E. Gallinucci); 0000-0002-0437-0725 (M. Golfarelli)

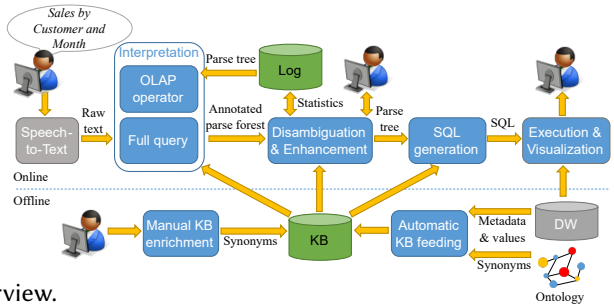


Figure 1: System overview.

In this paper, we describe COOL [4, 5] to convert natural language into *COnversational OLap* sessions composed of GPSJ queries and analytic operators. GPSJ [6] is the main class of queries used in OLAP. COOL is the first proposal addressing a full-fledged implementation for OLAP analytical sessions that is:

- *Automated and portable*: by reading metadata (e.g., hierarchy structures, measures, attributes, and aggregation operators) from a ROLAP engine, COOL automatically builds the minimal lexicon involved in the translation.
- *Robust* to user inaccuracies in syntax, OLAP terms, and attribute values, by exploiting metadata and implicit information.
- *Extendable* and easily configurable on a data warehouse (DW) without a heavy manual definition of the lexicon. The minimal lexicon is extendable by importing known ontologies in the Knowledge Base.

2. System overview

Figure 1 sketches a functional view of the architecture. Given a set of multidimensional cubes (DW), we distinguish between an offline phase (to initialize and configure the system) and an online phase (to enable the user interaction). We refer the user to [7] for an explanation of the DW terminology.

2.0.1. The offline phase

The offline phase automatically extracts the set of entities $\mathcal{E}$, i.e., the DW-specific terms used to express the queries. Such information is stored in the knowledge base (KB), which relies on the DFM expressiveness [7]. This phase runs *only* when the DW undergoes modification (e.g., cube schemas or data instances) and extracts all multidimensional metadata. A cube modeled as a *star schema* in a ROLAP engine consists of dimension tables (DTs: the cube hierarchies) and a fact table (FT: the cube). The `Automatic KB feeding` extracts measures (FT columns not in the primary key), attributes (DT columns), values (distinct instances of the DT columns), and hierarchies (either coded or inferred). These elements represent the lexicon necessary to translate natural language into conversational sessions. COOL supports lexicon extension with external synonyms that can be automatically imported from open data ontologies (e.g., Wordnet [8]) to widen the understood language. Besides the domain-specific terminology, the

KB also includes the set of standard terms that are domain independent and that do not require any feeding (e.g., group by, where, select). Further enrichment can be optionally carried out manually (e.g., when the physical names of tables/columns do not match a standard vocabulary).

2.0.2. The online phase

The online phase runs every time a natural language query is issued. In a hand-free scenario (e.g., [9]), the spoken query is initially translated to text by the `Speech-to-text` software module. Since this task is out of our research scope, we exploited a public Web Speech API. The uninterpreted text is then analyzed by the `Interpretation` step, refined in the `Disambiguation & Enhancement` step, translated by the `SQL generation` step, and finally executed and visualized by the `Execution & Visualization` step.

Interpretation consists of two alternative steps. `Full query` interprets the texts describing full queries (which happens when an analytic session starts). `OLAP operator` modifies the latest query when the user states an OLAP operator across a session (i.e., roll-up, drill-down, and slice&dice). The switch between the two steps to manage the conversation (i.e., a dialog between the user and COOL) is modeled by two states: *engage* (in which the system expects a `Full Query` to be issued) and *navigate* (in which the dialogue evolves by iteratively applying `OLAP operators` that refine the initial query). When COOL achieves a successful interpretation of a full query (i.e., it is able to run the query), it switches to the *navigate* state. Both `Full query` and `OLAP operator` follow these steps: (i) `Tokenization & Mapping`, (ii) `Parsing`, and (iii) `Checking & Annotation`.

Tokenization & mapping. A raw text T is a sequence of single words $T = \langle t_1, ..., t_z \rangle$. The goal of this step is to identify the entities in T, i.e., the only elements that will be involved in the `Parsing` step. Turning a text into a sequence of entities means finding a *mapping* between words in T and $\mathcal{E}$.

Definition 1 (Mapping & mapping function). *A mapping function $M(T)$ is a partial function that associates sub-sequences (or n-grams)[1] from T to entities in $\mathcal{E}$ such that: (i) sub-sequences of T have length n at most; (ii) the mapping function determines a partitioning of T; and (iii) a sub-sequence $T' = \langle t_i, ..., t_l \rangle \in T$ (with $|T'| \leq n$) is associated to an entity E if and only if $Sim(T', E) > \alpha$ (where $Sim()$ is a similarity function, later defined) and $E \in TopN(\mathcal{E}, T')$ (where $TopN(\mathcal{E}, T')$ is the set of N entities in $\mathcal{E}$ that are the most similar to T' according to $Sim(T', E)$). The output of a mapping function is a sequence $M = \langle E_1, ..., E_l \rangle$ on $\mathcal{E}$ that we call a* mapping.

The similarity function $Sim()$ is based on the Levenshtein distance and keeps token permutation into account to make similarity robust to token permutations (e.g., sub-sequences $\langle P., Edgar \rangle$ and $\langle Edgar, Allan, Poe \rangle$ must result similar).

Several mappings might exist between T and $\mathcal{E}$ since Definition 1 admits sub-sequences of variable lengths (corresponding to different partitionings of T) and associates the top similar entities to each sub-sequence. This increases the interpretation robustness since COOL chooses the best mapping through a scoring function. Given a mapping $M = \langle E_1, ..., E_m \rangle$, its score

[1] The term n-gram is used as a synonym of sub-sequence in the area of text mining.

⟨GPSJ⟩ ::= ⟨MC⟩⟨GC⟩⟨SC⟩ | ⟨MC⟩⟨SC⟩⟨GC⟩ | ⟨SC⟩⟨GC⟩⟨MC⟩ | ⟨SC⟩⟨MC⟩⟨GC⟩
| ⟨GC⟩⟨SC⟩⟨MC⟩ | ⟨GC⟩⟨MC⟩⟨SC⟩ | ⟨MC⟩⟨SC⟩ | ⟨MC⟩⟨GC⟩ | ⟨SC⟩⟨MC⟩ | ⟨GC⟩⟨MC⟩ | ⟨MC⟩

⟨MC⟩ ::= (⟨Agg⟩⟨Mea⟩ | ⟨Mea⟩⟨Agg⟩ | ⟨Mea⟩ | ⟨Cnt⟩⟨Fct⟩ | ⟨Fct⟩⟨Cnt⟩ | ⟨Cnt⟩⟨Attr⟩ | ⟨Attr⟩⟨Cnt⟩)+

⟨GC⟩ ::= "*group by*" ⟨Attr⟩+

⟨SC⟩ ::= "*where*" ⟨SCA⟩

⟨SCA⟩ ::= ⟨SCN⟩ "*and*" ⟨SCA⟩ | ⟨SCN⟩

⟨SCN⟩ ::= "*not*" ⟨SSC⟩ | ⟨SSC⟩

⟨SSC⟩ ::= ⟨Attr⟩⟨Cop⟩⟨Val⟩ | ⟨Attr⟩⟨Val⟩ | ⟨Val⟩⟨Cop⟩⟨Attr⟩ | ⟨Val⟩⟨Attr⟩ | ⟨Val⟩

⟨Cop⟩ ::= " = " | " <> " | " > " | " < " | " ≥ " | " ≤ "

⟨Agg⟩ ::= "*sum*" | "*avg*" | "*min*" | "*max*" | "*stdev*"

⟨Cnt⟩ ::= "*count*" | "*count distinct*"

⟨Fct⟩ ::= *Domain-specific facts*

⟨Mea⟩ ::= *Domain-specific measures*

⟨Attr⟩ ::= *Domain-specific attributes*

⟨Val⟩ ::= *Domain-specific values*

Figure 2: Backus-Naur representation of the `Full query` grammar. Entities from the KB are terminal symbols.

$Score(M)$ (i.e., the sum of entity similarities) is higher when M includes several entities with high similarity values. Intuitively, the higher the mapping score, the higher the probability to determine an optimal interpretation.

Parsing validates the syntactical structure of a mapping against a formal grammar and outputs a data structure called *parse tree* that is later used to translate a mapping into SQL.

In `Full query`, `Parsing` is responsible for the interpretation of a complete GPSJ query stated in natural language. A GPSJ query contains a measure clause (MC) and optional group-by (GC) and selection (SC) clauses. Parsing a full query means searching in a mapping the complex syntax structures (i.e., clauses) that build up the query. Given a mapping M, the output of the parser is a parse tree PT_M, i.e., an ordered tree that represents the syntactic structure of a mapping according to the grammar from Figure 2. To the aim of parsing, entities are terminal elements in the grammar.

For the sake of brevity, we omit the grammar of the `OLAP Operator` and we refer the user to [4]. Intuitively, our conversation steps are inspired by well-known OLAP visual interfaces (e.g., Tableau https://www.tableau.com/). To apply an OLAP operator, COOL must be in the state *navigate*, i.e., a full GPSJ query has been already successfully interpreted into a parse tree PT_C that acts as a context for the operator. The output of the `OLAP Operator` parser is a parse tree PT_M that is used to update PT_C.

Both GPSJ grammars are LL(1)[2] [10], not ambiguous (i.e., each mapping admits, at most, a single parse tree PT_M), and can be parsed by an LL(1) parser with linear complexity [10]. If the input mapping M is fully parsed, PT_M includes all the entities as leaves. Conversely, if only a portion of the input belongs to the grammar, an LL(1) parser produces a partial parsing, meaning that it returns a parse tree including the portion of the input mapping that belongs to the grammar (i.e., the PT rooted in ⟨GPSJ⟩). The remaining entities can be either singleton or

[2]Rules presented in Figure 2 do not satisfy LL(1) constraints for readability reasons.

complex clauses that could not be connected to the main parse tree. We will call *parse forest* PF_M the union of the parse tree with residual clauses. Obviously, if all the entities are parsed, it is $PF_M = PT_M$. Considering the whole forest rather than the simple parse tree enables ambiguities to be recovered in the `Disambiguation & Enhancement` step. Henceforth, we refer to the parser's output as a parse forest independently of the presence of residual clauses.

Disambiguation & enhancement. Due to natural language ambiguities, speech-to-text inaccuracies, and wrong query formulations, parts of the text can be misunderstood. The reasons behind the misunderstandings are manifold, including (but not limited to) a wrong usage of aggregation operators (e.g., summing non-additive measures), inconsistencies between attributes and values in selection predicates (e.g., filtering on product *"New York"*), or grouping by a descriptive attribute. Such parts of the parse forest are annotated as ambiguities. To reduce the parse forest PF_M to a single parse tree PT_M, the `Disambiguation & Enhancement` step solves ambiguities automatically whenever possible (by exploiting implicit information) or by asking appropriate questions to the user.

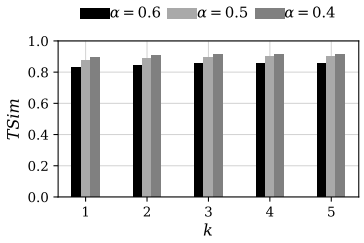
SQL generation translates a full-query parse tree into an executable SQL query. If an OLAP operator has been submitted, such tree must be updated accordingly. Given a full query parse tree, the generation of its corresponding SQL requires to fill in the SELECT, WHERE, GROUP BY and FROM statements. The SQL generation applies to both star and snowflake schemas [7] and is done as follows: SELECT (measures and aggregation operators from ⟨MC⟩ and attributes in the group by clause ⟨GC⟩); WHERE (predicates from the selection clause ⟨SC⟩); GROUP BY (attributes from the group by clause ⟨GC⟩); and FROM (measures, attributes, and values identify the fact and the dimension tables). The join path is identified by following the referential integrity constraints.

3. Experimental tests

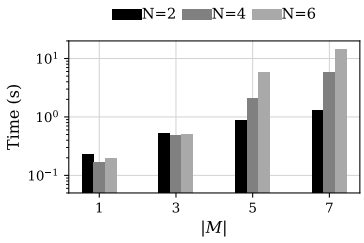
Tests are carried out on a real-world benchmark of analytics queries [11]. Since queries from [11] refers to private datasets, we mapped the natural language queries to the Foodmart schema[3]. The Automatic KB feeding populated the KB with 1 fact, 39 attributes, 12337 values and 12449 synonyms. Additionally, only 50 synonyms where manually added in the KB enrichment step (e.g., *"for each"*, *"for every"*, *"per"* are synonyms of the group by statement). While mapping natural language queries to the Foodmart domain, we preserved the structure of the original queries (e.g., word order, typos, etc.). Overall, 75% of the queries in the dataset are valid GPSJ queries, confirming how general and standard GPSJ queries are. The filtered benchmark includes 110 queries. We consider token sub-sequences of maximum length $n = 4$ (i.e., the [1..4]-grams) as no entity in the KB is longer than 4 words. Each sub-sequence is associated with the top N similar entities with similarity higher than α such that at least a percentage β of the tokens in T is covered. β is fixed to 70% based on an empirical evaluation. For the sake of brevity, only the major results are reported.

Effectiveness is evaluated as the tree similarity [12] $TSim(PT, PT^*)$ between the parse tree PT produced by our system and the correct one PT^*, which is manually written by us. Although only one parse forest is involved in the disambiguation phase; for testing purposes, it

[3] A dataset of food sales (https://github.com/julianhyde/foodmart-data-mysql).



(a) Effectiveness by varying the similarity threshold α and the $Top\text{-}k$ returned queries (with $N = 6$).



(b) Efficiency by varying the number $|M|$ of entities in the optimal tree and the top similar entities N (with $\alpha = 0.4$).

Figure 3: Empirical evaluation

is interesting to see if the best parse tree belongs to the top-k ranked ones. Effectiveness is more affected by the entity/token similarity threshold α and ranges in $[0.83, 0.89]$. In all cases, the best results are obtained when more similar entities are admitted and more candidate mappings are generated. Independently of the chosen thresholds the system results very stable (i.e., the effectiveness variations are limited) and even considering only one query to be returned its effectiveness is at the state of the art [2, 13, 14]. This confirms that (1) the choice of proposing only one query to the user does not negatively impact on performances (while it positively impacts on interaction complexity and efficiency) and (2) our scoring function properly ranks parse tree similarity to the correct interpretation for the query since the best ranked is in most cases the most similar to the correct solution.

As the previous tests do not include disambiguation, only 58 queries out of 110 are not ambiguous and produce parse trees that can be fed *as-is* to the generation and execution phases. Starting from the best configuration from the previous tests (for $N = 6$ and $\alpha = 0.4$), by applying an increasing number of correcting actions the effectiveness increases from 0.89 up to 0.94. Unsolved differences are mainly due to missed entities in the mappings.

As to efficiency, we ran the tests on a machine equipped with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz CPU and 8GB RAM, with the framework implemented in Java. Figure 3b shows the average execution time by varying $|M|$ and the number of allowed top similar entities N. The execution time increases with the number of entities included in the optimal parse tree, as such also the number of top similar entities impacts the overall execution time. We recall that effectiveness remains high also for $N = 2$ corresponding to an execution time of 1 second, raising to 10 seconds in the worst case.

4. Related works

Conversational business intelligence can be classified as a *natural language interface* (NLI) to *business intelligence* systems to drive analytic sessions. Despite the plethora of contributions in each area, to the best of our knowledge, no approach lies at their intersection.

NLIs to operational databases enable users to specify complex queries without previous train-

ing on formal programming languages (such as SQL) and software; a recent and comprehensive survey is provided in [15]. Overall, NLIs are divided into two categories: question answering and dialog. To the best of our knowledge, no dialog-based system for OLAP sessions has been provided so far. SQLizer [13] generates templates over the issued query and applies a "repair" loop until it generates queries that can be obtained using at most a given number of changes from the initial template. Domain-specific approaches add semantics to the translation process through domain-specific ontologies and ontology-to-database mappings. SODA [16] uses a simple but limited keyword-based approach that generates a reasonable and executable SQL query based on the matches between the input query and the database metadata, enriched with domain-specific ontologies. ATHENA [14] maps natural language into an ontology representation and exploits mappings crafted by the relational schema designer to resolve SQL queries. Analyza [17] integrates the domain-specific ontology into a "semantic grammar"(i.e., a grammar with placeholders for the typed concepts such as measures, dimensions, etc.) to annotate and finally parse the user query. Unfortunately, by relying on the definition of domain-specific knowledge and mappings, the adoption of these approaches is not plug-and-play as an ad-hoc ontology is rarely available and burdensome to create.

5. Conclusion

In this paper, we proposed COOL, a conversational OLAP framework supporting the translation of a natural language conversation into an OLAP session. COOL supports both the interpretation of GPSJ queries and OLAP operators.

Besides proposing a technical solution and a reference architecture, the contribution of the paper lies in the discussion of specific issues related to conversational OLAP systems. Since its conception, OLAP analysis aims to allow users to analyze data without requiring technological skills. Conversational OLAP represents a step forward in this direction. We believe that conversational OLAP can be particularly useful in the context of hand-free applications such as the ones we proposed in [3]: adding conversational capabilities to an augmented OLAP solution would be highly desirable whenever the user is working in the field (e.g., a warehouse or a factory) and is not in front of a computer. To close the loop, we are working towards enabling access to OLAP results in a conversational and hand-free fashion. The idea is to create *query summaries* (e.g., [18]) that can fit augmented-reality devices or (vocal) smart assistants.

References

[1] Y. Su, Towards Democratizing Data Science with Natural Language Interfaces, Ph.D. thesis, UC Santa Barbara, 2018.

[2] F. Li, H. V. Jagadish, Understanding natural language queries over relational databases, SIGMOD Record 45 (2016) 6–13. doi:10.1145/2949741.2949744.

[3] M. Francia, M. Golfarelli, S. Rizzi, A-BI^{+}: A framework for augmented business intelligence, Information Systems 92 (2020) 101520. doi:10.1016/j.is.2020.101520.

[4] M. Francia, E. Gallinucci, M. Golfarelli, COOL: A framework for conversational OLAP, Information Systems (2021) 101752. doi:10.1016/j.is.2021.101752.

[5] M. Francia, E. Gallinucci, M. Golfarelli, Conversational OLAP in action, in: Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021, OpenProceedings.org, 2021, pp. 646–649. doi:10.5441/002/EDBT.2021.74.

[6] A. Gupta, V. Harinarayan, D. Quass, Aggregate-query processing in data warehousing environments, in: Proc. VLDB, Morgan Kaufmann, San Francisco, CA, USA, 1995, pp. 358–369.

[7] M. Golfarelli, D. Maio, S. Rizzi, The dimensional fact model: A conceptual model for data warehouses, Int. J. Cooperative Inf. Syst. 7 (1998) 215–247. doi:10.1142/s0218843098000118.

[8] G. A. Miller, Wordnet: A lexical database for English, Commun. ACM 38 (1995) 39–41. doi:10.1145/219717.219748.

[9] M. Francia, M. Golfarelli, S. Rizzi, Augmented business intelligence, in: Proceedings of the 21st International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT Joint Conference, DOLAP@EDBT/ICDT 2019, Lisbon, Portugal, March 26, 2019, volume 2324 of *CEUR Workshop Proceedings*, CEUR-WS.org, Lisbon, Portugal, 2019, pp. 1–10.

[10] J. C. Beatty, On the relationship between LL(1) and LR(1) grammars, J. ACM 29 (1982) 1007–1022. doi:10.1145/322344.322350.

[11] K. Drushku, J. Aligon, N. Labroche, P. Marcel, V. Peralta, Interest-based recommendations for business intelligence users, Inf. Syst. 86 (2019) 79–93. doi:10.1016/j.is.2018.08.004.

[12] K. Zhang, D. E. Shasha, Simple fast algorithms for the editing distance between trees and related problems, SIAM J. Comput. 18 (1989) 1245–1262. doi:10.1137/0218082.

[13] N. Yaghmazadeh, Y. Wang, I. Dillig, T. Dillig, Sqlizer: query synthesis from natural language, PACMPL 1 (2017) 63:1–63:26. doi:10.1145/3133887.

[14] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, F. Özcan, ATHENA: an ontology-driven system for natural language querying over relational data stores, PVLDB 9 (2016) 1209–1220. doi:10.14778/2994509.2994536.

[15] K. Affolter, K. Stockinger, A. Bernstein, A comparative survey of recent natural language interfaces for databases, VLDB J. 28 (2019) 793–819. doi:10.1007/s00778-019-00567-8.

[16] L. Blunschi, C. Jossen, D. Kossmann, M. Mori, K. Stockinger, SODA: generating SQL for business users, PVLDB 5 (2012) 932–943. doi:10.14778/2336664.2336667.

[17] K. Dhamdhere, K. S. McCurley, R. Nahmias, M. Sundararajan, Q. Yan, Analyza: Exploring data with conversation, in: Proc. IUI, ACM, New York, NY, USA, 2017, pp. 493–504. doi:10.1145/3025171.3025227.

[18] M. Francia, M. Golfarelli, S. Rizzi, Summarization and visualization of multi-level and multi-dimensional itemsets, Information Sciences 520 (2020) 63–85. doi:10.1016/j.ins.2020.02.006.