Synthesis Lectures on Data Management

SYNTHESIS
COLLECTION OF TECHNOLOGY

Yunyao Li
Dragomir Radev
Davood Rafiei

Natural Language Interfaces to Databases

Springer

Synthesis Lectures on Data Management

Series Editor

H. V. Jagadish, University of Michigan, Ann Arbor, MI, USA

This series publishes lectures on data management. Topics include query languages, database system architectures, transaction management, data warehousing, XML and databases, data stream systems, wide scale data distribution, multimedia data management, data mining, and related subjects.

Yunyao Li · Dragomir Radev · Davood Rafiei

Natural Language Interfaces to Databases

Springer

Yunyao Li
Director of Machine Learning
Apple Experience Platform
San Jose, CA, USA

Dragomir Radev
Professor of Computer Science
Yale University
Newyork, NY, USA

Davood Rafiei
Professor of Computing Science
University of Alberta
Alberta, AB, Canada

ISSN 2153-5418 ISSN 2153-5426 (electronic)
Synthesis Lectures on Data Management
ISBN 978-3-031-45042-6 ISBN 978-3-031-45043-3 (eBook)
https://doi.org/10.1007/978-3-031-45043-3

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

To my husband, Huahai
my son Boyan
and my parents with love

—Yunyao Li

To my wife, Axinia,
my daughters, Laura and Victoria,
my parents,
and my students with love

—Dragomir Radev

To my wife, Malan
my children, Amin, Yasmin and Ali
and my parents with love

—Davood Rafiei

Foreword by the Series Editor

As humans, we manage a great deal of information: in our heads, as notes stuck on refrigerators, as shoeboxes full of receipts, and as files on our computers. To be able to find information relevant to any particular task, we need to have all our information organized, and we do, but usually in personal, and sometimes idiosyncratic, ways. As the amount of information grows, we need more structure.

This realization gave rise to the field of structured databases more than half a century ago. Today, most enterprise data, and much else, is stored in such structured relational databases. While such databases are excellent at correctly performing large computations and complete retrievals very efficiently, they are not easy to program or to query. A user not only needs to know a query language like SQL, but also needs to know precisely how the data have been structured and exactly what each attribute has been named.

This access impediment has been known right from the time relational databases were invented. Many smart people have devised many clever interfaces that make database access a little easier. But the holy grail has remained conversational natural language: we want to ask the computer questions in exactly the same way as we would ask questions to a human assistant.

However, even simple natural language queries remained elusive for decades. Only in the past dozen years or so, we have developed the ability to use natural language to pose database queries effectively. In other words, the field of Natural Language Interface for a Database transformed from being a distant vision to a vibrant research area with immediate practical value.

This book, written by three leaders in this field, presents an up-to-date survey of the state of the art today.

This Synthesis Lectures on Data Management series includes many works of importance for the field of databases, with their importance being derived from diverse dimensions. Some books cover a particularly novel research direction that a research group is pursuing. Others are important because they present an excellent survey of the state of the art in a topic of great interest. This particular monograph is a great example of the latter: it is a perfectly timed survey of a field that has enjoyed tremendous recent success. I hope you find it as illuminating as I do.

Ann Arbor, MI, USA H. V. Jagadish

The original version of the book has been revised: The author department details have been updated. The correction to the book can be found at https://doi.org/10.1007/978-3-031-45043-3_8

Preface

In the ever-expanding realm of data-driven exploration and decision-making, the ability to harness the power of natural language to interact with databases has become an indispensable pursuit. This book, "Natural Language Interfaces to Databases," presents a comprehensive journey through the fascinating world of Natural Language Interfaces to Databases (NLIDBs).

Chapter 1 serves as a gateway to this exciting topic, providing an Overview of NLIDBs. We begin with an interactive session with ChatGPT, providing a glimpse of the remarkable progress in this area and setting the stage for our discussion in the following chapters. We also highlight some of the pertinent challenges in deploying NLIDBs in real-world scenarios, which have driven the advancements we witness today.

In Chap. 2, we establish a foundation by exploring the steps in building an NLIDB, covering essential basics using examples. This chapter introduces the intricate anatomy of NLIDBs, shedding light on the essential components that enable communication between human language and databases. Readers will gain a holistic understanding of NLIDBs' underlying mechanisms.

Chapter 3, Data and Query Model, presents the key concepts in representing, querying, and processing structured data, as well as approaches for optimizing user queries. Our discussion covers various query languages, such as SQL and Relational Algebra, while highlighting their connections to first-order logic. By doing so, this chapter equips readers with sufficient knowledge of database systems, enabling them to navigate through the remaining content of the book with ease.

The process of transforming textual queries into structured form takes center stage in Chap. 4, Text to Data. This chapter provides a brief overview of early relevant work on semantic parsing and meaning representation before covering cutting-edge advancements in depth. Readers will gain a comprehensive understanding of key topics on text-to-data translation that empower NLIDBs to comprehend and interpret human languages.

Chapter 5 covers the essential topic of evaluation, where we present various evaluation methodologies, metrics, datasets, and benchmarks that play a pivotal role in assessing the effectiveness of mapping natural language queries to formal queries in a database and the overall performance of a system. Those metrics and benchmarks have been instrumental in advancing the field and providing us with the means to measure progress. By delving into the challenges of evaluating the equivalence between queries, readers will gain valuable perspectives on the continuous improvement of NLIDBs.

In Chap. 6, we explore the topic of data-to-text, where formal representations of data are transformed into running text in natural language. NLIDBs, equipped with structured data, can generate human-readable narratives, presenting results in a coherent and contextually relevant manner. Crafting narratives from data is a powerful tool with applications in various domains. This chapter comprehensively reviews domain-specific and domain-independent tasks that have been developed to evaluate the progress and improve the models in development.

Chapter 7 discusses Interactivity, the final frontier of NLIDBs. We introduce different dimensions of interactivity and the corresponding techniques for each dimension. We also discuss conversational NLIDBs and multi-modal NLIDBs where user input is beyond natural language. This chapter allows readers to have a comprehensive understanding of opportunities and challenges related to interactivity in NLIDBs.

Throughout the pages of this book, our endeavor has been to put together a balanced mixture of theoretical insights, practical knowledge, and real-world applications. Our earnest hope is that this compilation emerges as an invaluable resource, serving the interests of researchers, practitioners, and students eager to explore the fundamental concepts of NLIDBs.

In loving memory of Drago Radev, it is with a heavy heart and deep reverence that we present the final version of the book "Natural Language Interfaces to Databases." Our journey in crafting this work together spanned over more than two years, during which Drago's brilliance, unwavering dedication, and profound insights shaped every facet of its creation. His untimely passing in March 2023 was a profound loss, not only to those who had the privilege of knowing him personally but also to the academic and research communities worldwide. This book stands as a testament to his intellectual prowess and unyielding passion for knowledge dissemination. Although the completion of this work without his physical presence is bittersweet, we take solace in the knowledge that his spirit and ideas live on within these pages. It is our fervent hope that "Natural Language Interfaces to Databases" will continue to inspire, educate, and contribute to the advancement of our shared fields. May his legacy endure, and may this work serve as a tribute to the remarkable researcher who co-authored it—Drago Radev.

Let us embark on this journey of exploration into Natural Language Interfaces to Databases, which has become even more exciting with the rise of Large Language Models.

San Jose, CA, USA Yunyao Li
Alberta, Canada Davood Rafiei
July 2023

Acknowledgements

This book is made possible with the help and support of many people. H. V. Jagadish invited us to write this book. He also provided valuable comments on an earlier draft of the book. Dharaneeswaran Sundaramurthy and Christine Kiilerich from Spinger Nature pushed us to stay within our timeline. We wish to thank all of them for their help and support.

Many people helped us in putting together the content. In particular, Tao Yu reviewed some of the content as the writing was progressing and provided valuable feedback. He answered our questions or provided links when we needed. He has also contributed to the writing of Sect. 7.5. Aditya Parameswaran and Doris Lee have provided valuable input on Sect. 7.3. We also contacted many authors, especially when we had questions or wanted their permissions to use figures from their works, and we received an overwhelming number of positive responses. We wish to thank them for all the help and support we received.

We are also grateful to our the anonymous reviewer who, despite their busy schedules, has carefully read an initial draft of this book and provided detailed and constructive comments.

In addition to the aforementioned names who directly helped us in preparing the book, the joint work and discussions with our students and colleagues over the years helped shape the book. In particular, Davood wishes to thank Pirooz Chubak, Randy Goebel, Ehsan Kamalloo, Greg Kondrak, Haobin Li, Dekang Lin, Mahadi Hasan Nahid, Mir Tafseer Nayeem, Christopher Pinchak, and Mohammadreza Pourreza; Drago wishes to thank Catherine Finegan-Dollak, Jefferson Hsieh, Jonathan K. Kummerfeld, Victoria Lin, Alex Polozov, Peng Shi, Bailin Wang, Jason Wu, Rui Zhang, and Ruiqi Zhong. Yunyao wishes to thank Chris Baik, Ishan Chaudhuri, Laura Chiticariu, Ronald Fagin, Laura Haas, Eser Kandogan, Benny Kimelfeld, Rajasekar Krishnamurthy, Kun Qian, Sriram Raghavan, Lucian Popa, Frederick Reiss, Satinder P. Singh, Shivakumar Vaithyanathan, Huahai Yang, and Michelle Zhou.

Davood's research was supported by the Natural Sciences and Engineering Research Council of Canada and the University of Alberta. Drago's research was supported by the U.S. Natural Science Foundation and Yale University. Yunyao's research was supported by IBM Research, Apple, and Adobe.

Contents

1 Overview ... 1
1.1 A Session with ChatGPT ... 1
1.2 NLIDBs in the Wild ... 3
1.2.1 Language Support ... 5
1.2.2 Assisting Query Input ... 6
1.2.3 Expressivity ... 7
1.2.4 Customizability ... 8
1.3 What's Ahead ... 8
References ... 9

2 Building an NLIDB: The Basics ... 11
2.1 Example Database ... 11
2.2 Anatomy of NLIDB ... 12
2.3 Building an NLIDB ... 13
2.3.1 Query Understanding ... 14
2.3.2 Query Translation ... 18
2.3.3 External Knowledge ... 20
2.3.4 Interaction Generation ... 20
2.3.5 Result Generation ... 21
2.4 Summary ... 23
References ... 24

3 Data and Query Model ... 27
3.1 Conceptual Models ... 27
3.1.1 Entity-Relationship Model ... 28
3.1.2 UML ... 29
3.1.3 Ontology ... 30
3.2 Relational Model ... 30
3.2.1 Relational Query Languages ... 32
3.2.2 First-Order Logic ... 32

3.2.3 Relational Algebra 33
3.2.4 SQL 34
3.3 Graph Model 38
3.3.1 SPARQL 39
3.4 Storage and Indexing 41
3.5 Query Evaluation and Optimization 41
3.5.1 Query Parsing and Plan Generation 43
3.5.2 Query Optimizer 43
3.5.3 Evaluation Engine 45
3.6 Summary 46
3.7 Further Reading 46
References 46

4 Text to Data 49
4.1 Introduction 49
4.1.1 Natural Language Understanding and Natural Language Generation 50
4.1.2 Historical Overview 52
4.1.3 Semantic Parsing 54
4.2 Meaning Representations 55
4.2.1 First-Order Logic 55
4.2.2 Lambda Calculus 56
4.2.3 Abstract Meaning Representation 57
4.2.4 Word Embeddings 58
4.2.5 Semantic Compositionality 61
4.2.6 Knowledge Graphs and RDF 61
4.2.7 Syntactic Representations 62
4.2.8 Combinatory Categorial Grammar 64
4.3 Converting Sentences to Structured Form 66
4.3.1 Information Extraction 66
4.3.2 GeoQuery 67
4.3.3 Semantic Parsing 68
4.3.4 Semi-supervised Semantic Parsing 68
4.4 Neural Semantic Parsing 71
4.4.1 Sequence-to-Sequence Methods 71
4.4.2 Applications of Neural Semantic Parsing 73
4.5 Text-to-SQL 78
4.5.1 WikiSQL 81
4.5.2 Dataset Splits 82
4.5.3 Spider 82
4.5.4 Extensions to Spider 85

4.5.5 Selective Recent Papers ... 87
4.6 Summary ... 96
4.6.1 Further Reading ... 96
References ... 97

5 Evaluation ... 105
5.1 Methodology Overview ... 105
5.2 Datasets and Benchmarks ... 106
5.2.1 ATIS ... 107
5.2.2 GeoQuery ... 107
5.2.3 Scholar ... 107
5.2.4 Academic ... 107
5.2.5 Advising ... 107
5.2.6 IMDB and Yelp ... 108
5.2.7 Fiben ... 108
5.2.8 WikiSQL ... 108
5.2.9 SPIDER ... 109
5.2.10 BIRD ... 109
5.2.11 Benchmarks Statistics and Query Composition ... 110
5.2.12 Other Benchmarks ... 111
5.3 Reference-Based Evaluation ... 112
5.3.1 Generating a Reference ... 112
5.3.2 Candidate Answer Evaluation Based on a Reference ... 113
5.3.3 Performance Metrics ... 114
5.3.4 Semantic Equivalence ... 117
5.4 Human-Centric Evaluation ... 120
5.4.1 Expressing and Performing the Tasks ... 120
5.4.2 Quality of Generated Queries ... 120
5.4.3 Performance in Downstream Tasks ... 121
5.5 Other Performance Metrics ... 121
5.5.1 Resource Consumption ... 121
5.5.2 Query Hardness ... 122
5.5.3 Robustness to Noise and Ambiguity ... 123
5.5.4 Adaptability to Unseen Databases and Questions ... 124
5.6 Top Performing Models ... 126
5.6.1 The Models ... 126
5.6.2 Large Language Models ... 126
5.6.3 Constraining the Decoder ... 127
5.7 Further Reading ... 128
References ... 128

6 Data-to-Text 133
6.1 Introduction 133
6.1.1 Traditional Generation 134
6.1.2 Data-to-Text Generation 134
6.1.3 Abstract Meaning Representation (AMR) for Text Generation 136
6.1.4 Neural Generation 136
6.2 Domain Specific Table-to-Text 137
6.2.1 SRST 137
6.2.2 e2e 138
6.2.3 WebNLG 140
6.2.4 WikiBio 141
6.2.5 RotoWire 144
6.2.6 WikiTableT 148
6.3 Domain Independent Table-to-Text 148
6.3.1 ToTTo 148
6.3.2 DART 150
6.3.3 FeTaQA 151
6.3.4 TabFact 153
6.3.5 LogicNLG 153
6.3.6 Logic2Text 153
6.3.7 GEM 155
6.4 Pretraining for Tables 156
6.4.1 TURL 157
6.4.2 TUTA 157
6.4.3 TAPAS 158
6.4.4 TaBERT 160
6.4.5 Grappa 161
6.4.6 Tabbie 162
6.4.7 Other Recent Papers 163
6.5 Summary 169
References 170

7 Interactivity 177
7.1 Disambiguation 178
7.1.1 Ambiguity 179
7.1.2 Spell Correction 181
7.1.3 Interactive Disambiguation 183
7.2 Query Suggestion 187
7.2.1 Auto-Completion 187

7.2.2 Query Suggestions Beyond Auto-Completion 191
7.3 Automatic Data Insights 193
7.3.1 Categorization 194
7.3.2 Visualization Recommendation 197
7.4 Explanation 199
7.5 Conversational Natural Language Interfaces to Databases 202
7.5.1 Discourse Structure 202
7.5.2 Discourse Transition 204
7.5.3 Unidirectional Conversation 207
7.5.4 Bidirectional Conversation 210
7.6 Multi-modal Conversational NLIDB 213
7.6.1 Conversational Transition Modeling 215
7.6.2 Conversation via Query Suggestion 218
7.6.3 Discussions 219
7.7 Summary 220
7.8 Further Reading 220
References 221

Correction to: Natural Language Interfaces to Databases C1

Index 231

1 Overview

> If we are to satisfy the needs of casual users of data bases, we must break through the barriers that presently prevent these users from freely employing their native languages. – Ted Codd

This comment was made in the context of Rendezvous [1], a dialog-based system Ted Codd envisioned back in 1974, aimed at allowing casual users to effectively communicate with structured databases. This project was after he introduced the relational model for database management, which earned him the Turing Award. However, at the time, there was a total lack of resources for this project, with the field of natural language processing at its infancy and the scarcity of digitized text and computing resources. Huge progress has been made in the past 50 years, especially in the area of natural language processing and a significant reduction in the cost of computing resources. Such progress has pushed Codd's vision of Rendezvous to the forefront of current research and development.

This chapter introduces the subject using a short session with ChatGPT, showcasing the progress made in the area. This introduction is followed by a discussion of the challenges in deploying natural language interfaces to databases in the wild and what is being covered in the book.

1.1 A Session with ChatGPT

ChatGPT is a chatbot developed by OpenAI and is built on top of a family of Large Language Models (LLMs) including GPT-3.5 and GPT-4. The chatbot mimics human conversations using a set of large language models, each trained on some specific tasks. The use of those language models enables the chatbot to write essays, answer questions, compose music, and

Y. Li et al., *Natural Language Interfaces to Databases*, Synthesis Lectures on Data Management, https://doi.org/10.1007/978-3-031-45043-3_1

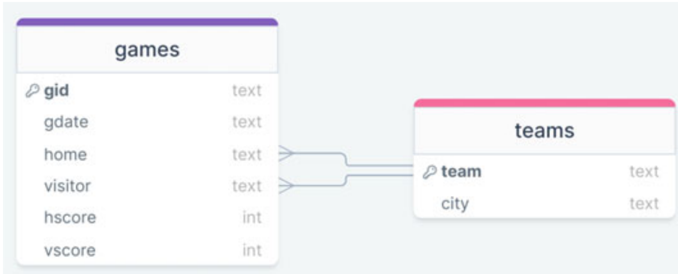

Fig. 1.1 An example database

write code. ChatGPT remembers previous prompts within the same conversation, making the conversation more natural.

Consider a database with schema shown in Fig. 1.1. The database has a table called *teams* that records the team names and their cities, and a table called *games* that records the games between different teams. Each game has a home team and a visitor team, and for each game both the home score and the visitor score, denoted by hscore and vscore respectively, are recorded.

Suppose we want to find out all teams that have played a game against *Oilers*, and we want to express the query in SQL. Having ChatGPT at our disposal, we give the table definitions and the query as a prompt to ChatGPT. Table 1.1 shows the prompt that is given and the chatbot response is shown in red. Without giving any additional information, the chatbot figures out that hscore refers to the score of the home team and vscore refers to the score of the visitor team, and it generates a correct SQL query. Our prompt ends the first table definition by a colon instead of a semicolon, but the chatbot simply ignores the typo.

We further type three more queries at the prompt, and the chatbot returns a correct SQL query in each case as shown in Table 1.2. A few observations that can be made here. First, the chatbot remembers the context, and it assumes that this is a continuation of the previous session, hence it uses the tables given earlier to write the query. Second, it is less sensitive to spelling mistakes in the input. For example, the query "*Write a SQLite query to teams* ..." is missing the term "*find*", but the chatbot still gives a correct SQL query. Third, the chatbot is less sensitive to the prompt formatting. Our last query starts with a query in English without the prefix given in the previous queries. The chatbot knows from the context that a SQL query is sought, and it provides that query. Finally, all generated SQL queries will compute correct results, though some generated SQL queries will give additional information that is not asked for. For example, the last two queries only ask for the team names, but the generated SQL queries will give not only the teams' names but also the number of their wins and losses. Also for the last query, the condition "losses = 0" is redundant.

ChatGPT is not developed in a vacuum. It builds upon many previous models and techniques in the field of natural language processing and code generation including the past work on natural language interface to databases (NLIDBs). Some key innovations that have

Table 1.1 Our session with ChatGPT (the model-produced response is in red)

```
–Tables
create table teams (
  team text,
  city text,
  primary key(team)
):
create table games (
  gid text,
  gdate date,
  home text,
  visitor text,
  hscore int,
  vscore int,
  primary key (gid),
  foreign key (home) references teams,
  foreign key (visitor) references teams
);
– Queries
Write a SQLite query to find teams that have played a game against Oilers.
```

```
SELECT DISTINCT visitor
FROM games
WHERE home='Oilers'
  UNION
SELECT DISTINCT home
FROM games
WHERE visitor='Oilers';
```

contributed to the development of ChatGPT include the progress in neural network architectures, such as transformer models, and the use of large amounts of training data to improve language understanding. Furthermore, techniques such as transfer learning and fine-tuning have helped the model to adapt to a wider range of domains and tasks.

1.2 NLIDBs in the Wild

With the major progress we are witnessing in NLIDB models especially LLM-based chatbots such as ChatGPT, it is expected that these new models will be incorporated in many commercial products and interfaces to databases. Microsoft has already started adding Chat-

Table 1.2 Continuing our session with ChatGPT (the model-produced responses are in red)

Write a SQLite query to find for each team that had a home game, the number of wins as a home team.
SELECT home, COUNT(CASE WHEN hscore > vscore THEN 1 END) AS wins FROM games GROUP BY home;
Write a SQLite query to teams that had more home wins than home losses.
SELECT home, COUNT(CASE WHEN hscore > vscore THEN 1 END) AS wins, COUNT(CASE WHEN hscore < vscore THEN 1 END) AS losses FROM games GROUP BY home HAVING wins > losses
Find teams that have won all their home games.
SELECT home, COUNT(CASE WHEN hscore > vscore THEN 1 END) AS wins, COUNT(CASE WHEN hscore < vscore THEN 1 END) AS losses FROM games GROUP BY home HAVING wins = COUNT(home) AND losses = 0;

GPT to its Office 365 products including Word and Powerpoint as well as to its Bing search engine [2]. There is no reason for not adding ChatGPT to Microsoft SQL server and other commercial database systems.

Even before the introduction of powerful LLM-based chatbots, an increasing number of commercial offerings have been supporting natural language queries in some form to enable business users to ask questions and query data in natural language. For instance, among the 20 vendors mentioned in the Gartner Magic Quadrant for Analytics and BI Platforms 2022, over half provide NLIDB support in some capacity. In this section, we discuss key aspects of NLIDB support with concrete examples from existing commercial offerings. Our goal is to help the readers to gain a holistic high-level understanding of key design decisions for NLIDBs in the wild and to acquire the necessary knowledge and framework to evaluate and compare such offerings on their own.[1]

[1] Our goal for this section is not to provide a comprehensive review of individual NLIDB in commercial offerings. We have selected the specific examples used in this section from online documentations of various commercial offerings for illustration purpose. The usage of an example from a particular commercial offering does not imply endorsement of the offering.

1.2.1 Language Support

We can characterize an NLIDB as supporting *controlled natural languages* (CNLs), if it restricts the grammar and vocabulary of its input, and supporting *(arbitrary) natural language*, if it does not. The benefit of using CNLs is to syntactically constrain the input to reduce or eliminate ambiguity and to reduce the complexity of understanding a full natural language [4]. Table 1.3 illustrates part of a comprehensive documentation on keywords and formats for a CNL. Many commercial offerings do not explicitly impose such constraints and opt to assist query input into queries that the systems can understand.

We can also categorize an NLIDB as *monolingual* or *multilingual*, depending on whether it supports one natural language only or not. Many NLIDBs in commercial offerings are

Table 1.3 Example Controlled Natural Language Support: SAP Search to Insight [3]

Query scope	What is supported	Examples
Chart a measure	Measure across a specified dimension's members Syntax: [Measure] by [Dimension]	`Show sales by region`
	Measure aggregated by a single dimension member Syntax: [Measure] for [Member]	`Show sales for John Smith`
	Specifying the dimension of a member Syntax: [Measure] for [Member] in [Dimension]	`Show sales for John Smith in sales representatives`
	Specifying multiple measures Syntax: [Measure] and / , [Measure] by [Dimension]	`Show sales and discounts by region` `show sales, discounts by region`
	Specifying multiple dimensions Syntax: [Measure] by [Dimension] and / , [Dimension]	`Show sales by region and priority` `Show sales by region, priority`
	Specifying multiple dimension members Syntax: [Measure] for [Member] and / , [Member]	`Show sales For Canada and USA` `Show sales for canada, USA`
Rank and Sort	Show top N values for the specified measure Syntax: top/highest/largest/best N [Measure] by [Dimension]	`Show top five Texas sales by priority`
	Show bottom N values for the specified measure Syntax: lowest/least/smallest/worst N [Measure] by [Dimension]	`Show lowest/least/smallest/worst five Texas sales by priority`

monolingual (usually English). Meanwhile, a few do provide multilingual support: Oracle Analytics, for instance, supports as many as 28 languages [5].

1.2.2 Assisting Query Input

NLIDB support in commercial offerings often provide a combination of the following common assistance features to speed up query formulation and to reduce input errors.

Autocomplete
: Autocomplete (also known as *typeahead*) predicts the rest of the user input as it is being typed. It guides the user to provide the keywords and terms that the system already understands. For instance, when a user types "*sales*" in the query, a NLIDB may provide a few possible ways to complete the query, all starting with the string "*sales*".

Autocorrect
: Autocorrect automatically corrects common spelling or typing errors and provides acronym and synonym matching. For instance, a NLIDB may automatically correct spelling error "*cafilornia*" to "*california*" and matches "*revenue*" to its synonym "*sales*".

Query Suggestion
: Query suggestions provide a list of possible queries that users can select from as they type to reduce query input efforts and help users find queries that the system can understand.

Query Recommendation
: Query recommendation suggests queries that are automatically determined or manually specified in advance for a given dataset. It facilitates the data exploration process by guiding users with limited knowledge of the system or the data with ideas of what types of queries they can ask as well as providing more experienced users with promising directions with interesting results or queries that may be hard to phrase.

We will discuss such features in more detail in Chap. 7.

1.2.3 Expressivity

We can categorize the expressivity of NLIDBs along the following dimensions.

Query Complexity Query complexity supported by an NLIDB depends on the type and the specific subset of query language(s) that it can translate questions into. The most popular underlying query languages includes SQL and SPARQL. Some systems support customized query languages specific to the underlying data store (e.g., Alibaba Cloud TableStore [6]). Some may support questions over multiple data stores with different query languages. Almost all systems support simple selection (e.g., "*show revenue*"), filtering (e.g., "*in US*"), aggregation (e.g., "*for the past 5 years*"), and ranking/sorting (e.g., "*by amount*"). However, more complex cases requiring nested query (e.g., "Show me countries sort by total of sales") are often not supported.

Data Function A NLIDB may allow users to specify other types of data functions besides queries. The data functions can be built-in functions or user-defined functions. Figure 1.2 shows an example question "what costs next 6 weeks" that corresponds to a built-in *forecast* function.

Visual Control A NLIDB may allow users to specify which visual to use for result display as part of the question.

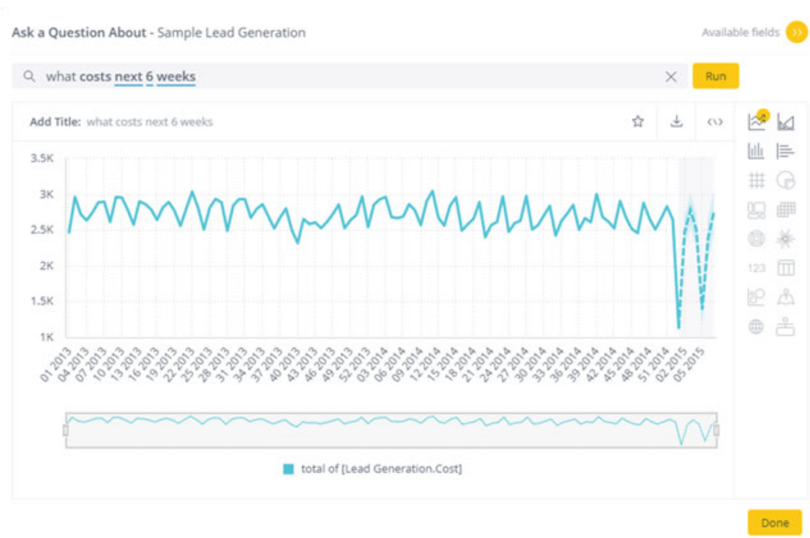

Fig. 1.2 Sample forecast query: Sisense simply ask