SQL Comprehension and Synthesis

George Rabeshi Obaido

School of Computer Science and Applied Mathematics,
University of the Witwatersrand, Johannesburg, South Africa.

A thesis submitted to the Faculty of Science,
in fulfilment of the requirements for the degree of
Doctor of Philosophy in Computer Science.

Supervisors:
Dr Hima Vadapalli
Prof Abejide Ade-Ibijola

22 May 2020

George Rabeshi Obaido
SQL Comprehension and Synthesis, 2020

SUPERVISORS:
Dr Hima Vadapalli[1]
Prof Abejide Ade-Ibijola[2]

SUPPORTED BY:
Council for Scientific and Industrial Research under the Department of Science and Technology in South Africa.

1 School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa.
2 Formal Structures, Algorithms and Industrial Applications Research Cluster, School of Consumer Intelligence and Information Systems, University of Johannesburg, Johannesburg, South Africa.

Dedicated to the
loving memory of my father, Mr Joseph Obaido.

DECLARATION

I declare that this thesis is my original and unaided intellectual work. This thesis is submitted for the degree of Doctor of Philosophy in Computer Science at the University of the Witwatersrand, Johannesburg. This work has not been submitted to any other University, or for any other degree.

George Rabeshi Obaido

May 2020.

ABSTRACT

Structured Query Language (SQL) remains the standard language used in Relational Database Management Systems (RDBMSs), and has found applications in healthcare (patient registries), businesses (inventories, trend analysis), military, education, etc. Although SQL statements are English-like, the process of writing SQL queries is often problematic for non-technical end-users in the industry. Similarly, formulating and comprehending written queries can be confusing, especially for undergraduate students. One of the pivotal reasons given for these difficulties lies with the simple syntax of SQL, which is often misleading and hard to understand. An ideal solution is to present these two audiences: undergraduate students and non-technical end-users with learning and practice tools. These tools are mostly electronic, and can be used to aid their understanding, as well as enable them write correct SQL queries. This work proposes a new approach aimed at understanding and writing correct SQL queries using principles from Formal Language and Automata Theory. We present algorithms based on: regular expressions for the recognition of simple query constructs, context-free grammars for the recognition of nested queries and a jumping finite automaton for the synthesis of SQL queries from natural language descriptions. As proof of concept, these algorithms were further implemented into interactive software tools aimed at improving SQL comprehension. Evaluation of these tools showed that the majority of participants agreed that the tools were intuitive and aided their understanding of SQL queries. These tools should, therefore, find applications in aiding SQL comprehension at higher learning institutions and assist in the writing of correct queries in data-centered industries.

PUBLICATIONS

Some portions of this thesis have been published in the following articles. These portions include figures, algorithms, equations and descriptions of concepts. These papers are:

[1] Abejide Ade-Ibijola and George Obaido (2017). *S-NAR: Generating Narrations of SQL Queries using Regular Expressions*. In the ACM Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT), pp 11-18, Bloemfontein, Free State. ISBN: 978-1-4503-5250-5. URL: https://dl.acm.org/citation.cfm?doid=3129416.3129454. [South Africa].

[2] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli (2018). *Generating SQL Queries from Visual Specifications*. In Communications in Computer and Information Science (CCIS), vol. 963, pp 315–330, Springer, Cham, Switzerland. ISBN: 978-3-030-05813-5. URL: https://link.springer.com/chapter/10.1007/978-3-030-05813-5_21. [Switzerland].

[3] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli (2019). *Generating Narrations of Nested SQL Queries using Context-free Grammars*. In the proceedings of IEEE Conference on ICT and Society (ICTAS), pp 13:1–6, 6th to 9th March, Durban. URL: https://ieeexplore.ieee.org/document/8703620. [South Africa].

[4] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli (2019). *TalkSQL: A Tool for the Synthesis of SQL Queries from Verbal Specifications*. In the proceedings of the IEEE International Multidisciplinary Information Technology and Engineering Conference (IMITEC), pp 469–478, November 21st to 22nd, 2019. [South Africa].

[5] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli (2019). *Synthesis of SQL Queries from Narrations*. In the proceedings of the 6th IEEE International Conference on Soft Computing & Machine Intelligence (ISCMI), pp 195–201, ISBN: 978-1-7281-4576-1, November 19th to 20th, 2019. [South Africa] — ***Best Presentation Award***.

Other articles on related projects:

[6] Abejide Ade-Ibijola and George Obaido (2019). *XNorthwind: Grammar-driven Synthesis of Large Datasets for DB Applications*. In International Journal of Computer Science (Scopus), Vol. 46, Issue 4, pp 541–551, International Association of Engineers (IAENG). URL: http://www.iaeng.org/IJCS/issues_v46/issue_4/IJCS_46_4_05.pdf. [Hong Kong].

ACKNOWLEDGMENTS

The work presented in this thesis has been a life-changing experience and it would not have been possible without the support and guidance that I have received from many people. I, therefore, take this opportunity to extend my sincere gratitude and appreciation to all those who have made this PhD thesis possible.

Firstly, I would like to acknowledge my indebtedness and render my warmest thanks to my supervisors, Prof Abejide Ade-Ibijola and Dr Hima Vadapalli. I am grateful to Prof Abejide for suggesting my topic as well as for introducing me to the exciting fields of Formal Language and Automata applications. He has taught me, both consciously and unconsciously, the elements of how good research should be done. He has spent countless hours editing my work in addition to constantly reminding me of how we needed to 'flush' accepted manuscripts; words which I will fondly remember. I will forever value all the time, advice, inspiration, encouragement and ideas that he has contributed to my PhD pursuit. The joy and enthusiasm that he has for research is truly remarkable and this has been a motivational factor throughout my PhD journey. Similarly, I would like to extend my appreciation and gratitude to Dr Hima for her constant encouragement, contribution and financial support towards the conferences that I was able to attend and at which the research papers found in this work were presented.

I am indebted to the Council for Scientific and Industrial Research - Department of Science and Technology of South Africa, under the Interbursary MSc/PhD bursary support programme for funding this research. I am also grateful to the Association for Computing Machinery, and Black in Artificial Intelligence, especially Rediet Abebe and Timnit Gebru, for providing travel grants to the United States of America and Canada respectively.

I would also like to extend my thankfulness to Prof Cajethan Iheka, Dr Sihlobosenkosi Mpofu, Dr Sarah Taylor, Dr Tarisai Rukuni, Abiodun Modupe, Michelle van Heerden and Dr Amos Anele for their words of encouragement during the writing of this thesis. My sincere appreciation goes out to Cynthia Lawrence, Haroon Motalib, Divashni Naicker and Jowilna Storm for offering me a part-time lecturing position at Pearson Institute of Higher Education, South Africa. In addition, I am grateful to my students, Laruschka Smit, Angelique Hassett, Jason Petrie, Rose Nnanna and Prisca Bifutuka, who have often assisted with crowdsourcing tasks, which has contributed to the completion of my research.

My appreciation also goes out to Jude Oyasor, Kehinde Aruleba, Blessing Ogbuokiri, Elvis and MaryAnn Umejiaku, Iyiola Ogunrekun, Michael Ayawei, Mustapha Oloko-Oba, Emmanuel Mudakiri, Segun and Bukola Adebayo, Dennis Uzizi, Oluwatobi Onawole, Izuchukwu Ohaegbu, Mbulelo Mthembu, Wainright Acquoi, Lance Krasner, Alan Michael, Ndivhuwo Nemudiviso and Senyo Cudjoe, who are my colleagues and friends for life, for making this journey exciting and fun.

Lastly and most importantly, I would like to thank my family for all their love and encouragement. I am thankful to my only surviving parent, my mother Margaret, who has constantly supported me in all of my academic pursuits. My appreciation goes to my in-laws, Melfie (mother), Rejoice (sister) and Whitney (niece) for constantly reminding me that I needed to rest throughout this journey. Many thanks to my siblings, Christiana, Anthony, Victoria, Sarah and Faith, for their unflinching support. And most of all, to my loving wife Thobekile, for her love and patience with me during those late nights spent in the research lab.

PREAMBLE

This PhD thesis investigates the problem of learning and writing correct SQL queries. It presents a number of approaches to the problem of SQL comprehension and synthesis. As a whole, it covers the application of the formal language and automata theory to the problem of SQL comprehension and synthesis. This preamble, by way of an introduction, provides the contributions and organisation of this thesis. It shows the list of related domains with keywords and highlights non-academic talks that have been presented on this work.

Technical contributions.
The technical contributions of this work are as follows:

1. *Formal Language and Automata Theory*: new formalisms for recognising SQL queries to generate *narrations* using regular expressions (REs) and context-free grammars (CFGs), and a jumping finite automaton (JFA) for synthesising SQL queries from natural language specifications have been presented.

2. *Software Prototypes*: five software tools are presented. Firstly, `S-NAR` for assisting end-users to understand simple SQL queries [Ade-Ibijola and Obaido 2017]; secondly, a tool called the `SQL Narrator` for assisting end-users to understand nested SQL queries [Obaido *et al.* 2019a]; thirdly, `Narrations-2-SQL` for translating natural language descriptions into SQL queries [Obaido *et al.* 2019b]; fourthly, the `SQL Visualiser` that uses visual specifications to build a query [Obaido *et al.* 2018], and finally, `TalkSQL` aimed at assisting end-users to understand SQL queries using speech inputs [Obaido *et al.* 2019c].

3. *Evaluation of Prototypes*: The usefulness of these software tools was evaluated and the evaluation results are presented.

Thesis organisation.
This thesis is organised into four parts. Part i contains the introduction, definition of terms and a review of literature relevant to this study. Part ii and Part iii presents the major contributions of this work. Part iv evaluates the developed tools and presents the conclusions and future directions.

Domain of research.
The following categories shows the 2012 ACM[3] CCS[4] that this research is based on.

- Theory of computation, formal languages and automata theory, and grammars and context-free languages.

- Applied computing, computer-assisted instruction and interactive learning environment.

- Computers and education, computer and information science education and computer science education.

3 Association for Computing Machinery
4 Computing Classification System

- Computing methodologies, artificial intelligence and natural language processing.

Keywords.
The keywords used in this research are as follows:

- SQL comprehension, SQL query narration, SQL tutoring, intelligent tutoring system, regular languages, context-free grammar, jumping finite automaton, query by speech, visual specifications, verbal specifications, natural language processing, learning via visualisation, learning via narrations, language translation, relational database and synthesis of things.

Non-academic Talks.
Some ideas of this thesis was presented at the following event.

- Doctoral Consortium of the ACM FAT*[5] Conference in Atlanta, Georgia in the USA on the 29th January, 2019.

5 Fairness, Accountability and Transparency

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

Table 1: List of Acronyms

AI	Artificial Intelligence	NLG	Natural Language Generation
API	Application Programming Interface	NLIDB	Natural Language Interfaces to Database
BI	Business Intelligence	NLP	Natural Language Processing
BNF	Backus-Naur Form	NLQ	Natural Language Query
CFG	Context-free Grammar	NLTK	Natural Language Toolkit
CLT	Cognitive Load Theory	NLU	Natural Language Understanding
CNN	Convolutional Neural Network	NMT	Neural Machine Translation
CS	Computer Science	PoS	Part-of-Speech
DB	Database	PBL	Problem Based Learning
DBN	Deep Belief Network	QA	Question Answering
DDL	Data Definition Language	RDBMS	Relational Database Management System
DML	Data Manipulation Language	RE	Regular Expression
EBNF	Extended Backus-Naur Form	RNN	Recurrent Neural Network
ERD	Entity Relationship Diagram	SFA	Syntax-free Approach
FLA	Formal Language and Automata Theory	SMT	Statistical Machine Translation
GUI	Graphical User Interface	SPARQL	SPARQL Protocol and RDF Query Language
HCI	Human-Computer Interaction	SQL	Structured Query Language
HMM	Hidden Markov Model	SSIS	SQL Server Integration Service
IBM	International Business Machines	SVM	Support Vector Machine
ITS	Intelligent Tutoring System	UD	Universal Dependency
JFA	Jumping Finite Automaton	UI	User Interface
MT	Machine Translation	WER	Word Error Rate

Part I

INTRODUCTION AND BACKGROUND

Structured Query Language (SQL) is the *de facto* query language for most relational databases, containing commands used to access and manipulate data. Since released by Codd [1970], SQL has been used by most database vendors for their products. Although SQL is highly declarative, many end-users encounter several challenges in writing correct queries. Such challenges have limited its use as the preferred database language of choice. This part of the thesis provides an introduction to what SQL is, describes several terms used, and reviews literature similar to this work.

This part contains three chapters. Chapter 1 introduces and provides the context for this research. Chapter 2 outlines several definitions that were used in this study. Chapter 3 presents the literature on SQL comprehension and other areas of study.

1 INTRODUCTION

For the past three decades, Structured Query Language (SQL) has been the preferred database language for relational database management systems (RDBMSs) [Kawash 2014; Heller 2019a]. Since being adopted as an ANSI[1] and ISO[2] standard, SQL has been widely used by most database vendors for their commercial products, such as IBM[3] DB2, Microsoft SQL Server and Access, SAP HANA, Splunk DB, Teradata DB, etc [Levene and Loizou 2012; Bonham-Carter 2014; Heller 2019b]. Similarly, many open-source RDBMSs have been introduced that support SQL, such as Oracle's MySQL, PostgreSQL, Mozilla's Firebird, MariaDB, Ingres, etc [Soflano *et al.* 2015; Heller 2019b].

SQL has found many applications in academia and industry [Borodin *et al.* 2016; Chamberlin 2012; de Silva 2017]. In higher learning institutions, SQL is taught as part of the introductory database course in the undergraduate curriculum [Silva *et al.* 2016]. Learning SQL is a pivotal skill that a Computer Science (CS) student ought to master as it is pertinent for an entry role in many diverse industries [Cappel 2002; Sander and Wauer 2019]. Garner and Mariani [2015] suggest that even non-technical end-users, such as financial managers, stock brokers and controllers as well as HR managers in industry, should be able to write queries as part of their job functions. However, that may not always be the case. Figure 1 shows the interaction between end-users and a RDBMS.

It is worth noting that SQL underpins a range of applications and programming languages to allow users to manipulate and retrieve information. These applications range from e-commerce, Internet of Things (IoT), commercial as well as open-source software. The following examples show some applications of SQL:

1. A number of Extract Transform Load (ETL)[4] tools such as SQL Server Integration Services (SSIS), Skyvia and Informatica use SQL to communicate with databases.
2. Programming languages such as Python and PHP usually embed SQL in their query strings when connecting to a database.
3. Top Business intelligence (BI) tools such as the Microsoft Power BI and Tableau use SQL to create reports and charts while working with data.
4. Most mobile application development frameworks: hybrid (Ionic, PhoneGap, Xamarin, etc) and native (Android Studio and Swift) support SQL in their engines; since many of these frameworks support SQLite.
5. Big data analytic tools that support the RESTFUL API such as Elasticsearch and Sphinx execute SQL queries to produce results.

Just like any other language, it is generally agreed that SQL is challenging for non-technical end-users and undergraduate students alike [Bider and Rogers 2016; Grillenberger and Brinda 2012; Folland 2016]. Many studies have identified the difficulties faced by

1 American National Standards Institute
2 International Organization for Standardization
3 International Business Machines
4 A data warehousing process for extracting data from different sources to fit into organisational needs

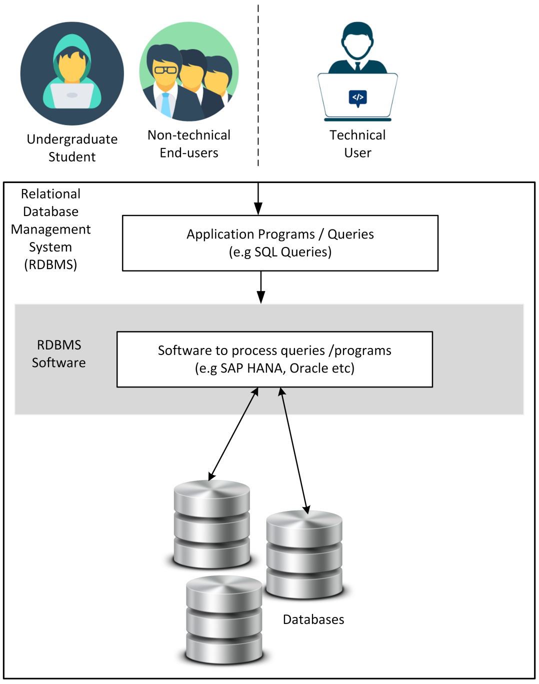


Figure 1: An interaction between users and a RDBMS

non-technical end-users when writing SQL queries [Garner and Mariani 2015; Najar *et al.* 2016; Atchariyachanvanich *et al.* 2017]. These difficulties include: the burden of memorising database schemas, its declarative nature, and the naive perception that SQL is easy. Li and Jagadish [2014b] acknowledge that non-technical end-users struggle to comprehend queries written by technical users [Li and Jagadish 2014b]. Similarly, Garner and Mariani [2015] noted that this is also true for undergraduate CS students. They emphasise that students struggle to learn SQL alongside a procedural or object-oriented programming language. Another study by Ahadi *et al.* [2015] highlighted that simple and nested SQL queries are often problematic for students. Hence, it has become imperative for researchers to design learning and practice aids (mostly electronic) for these two audiences – undergraduate students, and non-technical industry end-users – to aid their comprehension of SQL. Moreover, tools that provide brief explanations of a query's functionality and visualisation of a query's output, can assist these users in understanding SQL [Kokkalis *et al.* 2012; Danaparamita and Gatterbauer 2011]. In addition, tools with speech recognition capabilities can assist visually impaired learners in understanding SQL [Berque *et al.* 2003; Mealin and Murphy-Hill 2012].

This work introduces "SQL Comprehension and Synthesis", using approaches that allow users (students and non-technical end-users) to understand SQL queries and provide interfaces to write queries in clear English terms – in a natural language. Such approaches abstract user-level queries and provide granularity of representation, identify syntax and semantic errors and provide possible solutions. Query comprehension involves tasks that make use of a mental process aimed at query reading and explanation [Shneiderman 1978; 2000]. Although query comprehension has been investigated as *patterns* introduced to aid knowledge transfer by Faroult and Robson [2006], it was specifically targeted for experienced database developers to solve more complex query tasks. However, if a novice user cannot even construct simple queries, these patterns tend to offer no solution to the current challenge. Compared to the program comprehension domain that has gained popularity in recent years [Storey 2006; Ade-Ibijola *et al.* 2014; 2015], there are still many unexplored areas of research in the SQL comprehension domain.

Over the past decades, the program comprehension domain has recorded many successes. This is motivated by two classic cognitive theories, namely the *top-down* and *bottom-up* [Brooks

1977; Storey 2006] approaches. The top-down approach focuses on how programs are perceived by programmers based on a series of hypotheses. The bottom-up approach uses tools to aid the comprehension process. This process was termed chunking, which shows obvious parts of code that a programmer may recognise. Together, these theories are regarded as the unified model [Von Mayrhauser and Vans 1993]. Within Computer Science Education, a systematic approach was introduced by Fincher [1999] aimed at teaching programming without the syntax of the language. Such an approach is regarded as the Syntax-free Approach (SFA). The SFA has been applied in many program comprehension problems in elementary education [Mannila *et al.* 2014], middle-age education [Grover *et al.* 2015], and for undergraduate programs [Lahtinen *et al.* 2005; Ade-Ibijola *et al.* 2014; 2015; Ade-Ibijola 2016b]. Therefore, this was the approach employed in this research to assist students and non-technical end-users understand SQL queries.

1.1 PROBLEM DESCRIPTION

This section describes the problems that were solved in this thesis. They are listed under the following categories.

1.1.1 *Generating Narratives for Simple Queries*

Many tools have been developed to assist end-users understand SQL queries [Cembalo *et al.* 2011; Folland 2016]. The majority of these tools apply visualisation and interactive techniques to aid the comprehension of SQL queries. While these tools have shown to be effective at improving comprehension, many end-users struggle to understand the constructs and underlying logic behind SQL. Given this difficulty, it is important to granulate these queries into a form that is free from SQL syntax. With respect to this, we have answered these questions:

1. How else can we abstract queries so that they are easily understood by students and non-technical end-users?
2. Can we design a tool that addresses this problem?

To answer these questions, we took a cue from the program comprehension domain. Within the programming pedagogy, Fincher [1999] suggested that if teaching a programming language with the syntax affects the learning process, teaching without it would attempt to avoid it. In her words, this is "paradoxical" [Fincher 1999]. The author suggested that the ideal way of teaching a programming language was the SFA, which abstracts the syntax of the language into a readable form. That is, translating programs back into syntax-free algorithms specified in natural language descriptions known as "narrations" [Ade-Ibijola *et al.* 2014]. Hence, we *propose a tool that generates narratives for simple SQL queries using Regular Expressions (REs) for SQL comprehension*.

1.1.2 *Generating Narratives for Nested Queries*

A number of studies have identified that nested queries pose great difficulties for end-users [Woolf 2010; Ahadi *et al.* 2016]. In particular, end-users struggle to understand nested query constructs using the `SELECT` commands and `GROUPBY` clauses. An end-user who struggles to understand a simple SQL query will surely find nested queries challenging. Many tools have been built to handle simple queries, hence there is a growing demand for tools that aid nested SQL query comprehension [Neumann and Kemper 2015]. It has become attractive for researchers to develop tools that handle these types of queries. We have answered the following questions:

1. How can we abstract nested SQL queries so that users can understand them?
2. Can we develop a tool for this?

Given these questions, we *extended the use of narrations to describing nested queries using a Context-free Grammar (CFG) for nested SQL query comprehension*. Since nested queries appear in more complex forms, REs are not suitable to handle this hitch as they are only useful for lexical analysis.

1.1.3 *Synthesising SQL Queries from Narrations*

In the business sector, many professionals such as financial analysts, marketing executives, stock brokers, and mining experts use DB applications on a daily basis. These end-users often are unable to write correct queries. In most cases, they can clearly describe the intended task but lack the ability to formulate a correct query. As a result, they seek help from online sources and from technical experts to assist them with their query operation [Yaghmazadeh *et al.* 2017a]. Zhang and Sun [2013] identified two popular approaches to assist end-users in writing SQL queries: Graphical User Interfaces (GUIs) and the use of programming languages. It is worth noting that most RDBMSs are designed with GUI features that an end-user may struggle to find. Even so, writing SQL alongside a programming language requires good technical skills. Since end-users lack good programming skills, they may struggle to write correct SQL queries [Folland 2016; Prior 2014]. An ideal approach would be to allow these users to express their requests in natural language. We have used the term "narrations" to describe a natural language. The following questions have been answered:

1. How can we synthesise SQL queries from narrations?
2. Can we describe an approach that automatically synthesises these queries?
3. Can we develop a tool for this?
4. What are end-users' perceptions of the tool?

To answer this question, an approach that considers a context-sensitive language such as a natural language will suffice. Since natural languages are ambiguous, we *propose the use of a Jumping Finite Automaton (JFA) to translate natural language descriptions into SQL queries*.

1.1.4 *Generating SQL Queries using Visual Specifications*

Many RDBMSs contain query builders which are used to visualise a database schema through multiple mouse clicks [Marcus *et al.* 2019]. Query builders allow a user to specify a query which retrieves multiple columns and table relations from a database. Whilst this approach provides many advantages, end-users need to remember that memorising a database schema may be difficult [Garner and Mariani 2015]. In some cases, syntax errors from RDBMSs may be difficult for end-users to debug and the feedback received may not offer much help [Lavbič *et al.* 2017]. In addition, the SQL `SELECT` command is often problematic for learners [Sadiq *et al.* 2004]. In relation to this, we have answered the following questions:

1. Which visualisation would assist users understanding of SQL queries?
2. Does our approach require knowledge of SQL?
3. Can we compare our approach with other existing tools?

To answer these questions, we *propose an image query visualiser that uses the drag and drop interactions to generate a SQL query*.

1.1.5 *Generating SQL Queries using Verbal Specifications*

Many Intelligent Tutoring Systems (ITSs) use speech commands to provide immediate and customised instruction to serve both educational and industrial needs [Graesser *et al.* 2012].

Whilst these speech ITSs perform operations seamlessly, they only consider the SELECT statement, ignoring other query commands [Garner and Mariani 2015]. In addition, some speech ITSs fail to provide comprehensive feedback to a user. The following questions have been answered:

1. Can we build a tool that supports speech inputs?
2. Can visually impaired users use this tool?

To answer these questions, we *propose a speech-based system that takes speech inputs from a user, converts these into a query and provides speech, textual and visual feedback to the user*.

1.2 RESEARCH CONTEXT

1.2.1 *Aim and Objectives*

The aim of this research is to develop a new approach that makes it easier to assist non-technical end-users and students to understand SQL queries and also build software tools that tests this new approach. The objectives that make up the aim are to:

1. *aid the comprehension of queries through narrations*. The term "narrations" was first coined by Ade-Ibijola *et al.* [2014] used to describe a textual approach aimed for novice program comprehension. This textual approach describes a program's functionality free from a program's syntax, which is written in plain text. This idea is based on the SFA idea by Fincher [1999]. In this work, we have extended the use of narrations to describe simple and nested queries. This will be based on REs and CFGs, two formal language techniques for generating languages.

2. *translate natural language specifications of queries into standard SQL queries*. In most cases, non-technical end-users struggle to write correct SQL queries. This aspect of the research focuses on assisting non-technical end-users to write SQL queries. It allows end-users to specify their request using a natural language, which undergoes a number of transformations before a query is generated. This uses a JFA, an automata algorithm.

3. *build an interactive visualiser*. Visualisation has shown to improve the cognitive workload for understanding a concept [Cembalo *et al.* 2011; Mitrovic and Ohlsson 2016]. The visualisation we have employed uses images depicting SQL operations to build queries. This approach uses the 'drag and drop' interactions for generating SQL queries.

4. *build a speech to SQL query synthesiser*. This is meant to assist users understanding of queries using speech inputs. This aspect also employs REs for the recognition of queries for feedback generation.

5. *evaluate the impact of the comprehension aids proposed*. An online survey will be used to determine the effect of this tools.

1.2.2 *Scope*

This research is based on the assumptions that:

1. the scope is confined to the use of formal language techniques using REs and CFGs for the recognition of SQL queries, and an automata-based algorithm using a JFA for the automatic synthesis of SQL queries from natural language specifications.

2. the focus will be on comprehending only SQL queries in simpler and nested forms.

3. we only focus on comprehending and synthesising SQL queries for these end-users, namely: *undergraduate students* in higher learning institutions and *non-technical users* in the business sectors.

1.2.3 *Questions*

The questions of interest that have been answered in this research are as follows:

1. *can we build tools that can automatically narrate an SQL query?* – yes, we answered this question in Part ii. In this part, we developed two tools; using REs to recognise simple SQL queries (in Chapter 4) and a CFG for nested SQL query recognition (in Chapter 5). The resultant narrations were presented in these chapters.

2. *given good narrations, is it possible to come up with a valid SQL query?* – yes, we showed that this was possible. We used a JFA to recognise natural language descriptions and used algorithms to generate a valid query. The idea is presented in Chapter 6 under Part iii.

3. *can visually impaired users use these tools?* Definitely, they can use these tools. We showed that with the aid of a conversational tool in Chapter 7 in Part iii, that these type of learners can use these tools to manipulate and access data from a database.

4. *can these tools find applications in both academic and business environments?* This question is answered in Chapter 9 in Part iv. Participant feedback is also presented here.

1.3 TECHNICAL CONTRIBUTIONS

The contributions of this work are divided into three categories: formal techniques for the comprehension and synthesis of SQL queries, software prototypes of these techniques and evaluation of the prototypes.

1.3.1 *Formal Language and Automata Theory*

Formal language and automata theory (FLA) has been used in a wide spectrum of application areas. This research explored the ideas from this domain for SQL understanding. In Part ii and Part iii, we have:

1. designed REs, a class of regular languages for the recognition of SQL query constructs,

2. designed a CFG, a subset of irregular languages for the recognition of nested SQL queries, and

3. used a JFA, an automata-based algorithm, to recognise natural language specifications.

1.3.2 *Software Prototypes*

The FLA algorithms using REs, CFG and JFA were implemented into a number of software prototypes. The following prototypes are:

1. `S-NAR` [Ade-Ibijola and Obaido 2017]: We developed a tool that uses REs which automatically generates a narration from a query in order to aid the comprehension of SQL queries.

2. `SQL Narrator` [Obaido *et al.* 2019a]: This aspect is an improvement of `S-NAR` that describes the automatic generation of narrations from nested SQL queries using a CFG.

3. `Narrations-2-SQL` [Obaido *et al.* 2019b]: We presented a tool that uses a JFA for the recognition of natural language specifications of queries and translated them into a SQL query.

4. `SQL Visualiser` [Obaido *et al.* 2018]: We used visual specifications that represent SQL commands to build queries.

5. TalkSQL [Obaido *et al.* 2019c]: We presented a speech-based system that takes speech inputs from a user and uses REs to convert this into a query.

1.3.3 *Evaluation of Prototypes*

For each aspect of the study, we present the results of the evaluation of the tools that were developed in Part iv.

1. S-NAR [Ade-Ibijola and Obaido 2017] was tested on 5000 queries and reported an accuracy of 96%.
2. SQL Narrator [Obaido *et al.* 2019a] reported that 98.1% agreed that the tool enabled them understand nested queries.
3. Narrations-2-SQL [Obaido *et al.* 2019b] showed that 96.9% agreed the tool would be helpful to end-users.
4. SQL Visualiser [Obaido *et al.* 2018] reported that 92.16% indicated that the tool aided their understanding of SQL syntax.
5. TalkSQL [Obaido *et al.* 2019c] concluded that 87.6% acknowledged that the tool will help visually impaired learners to write correct queries using speech inputs.

1.4 THESIS ORGANISATION

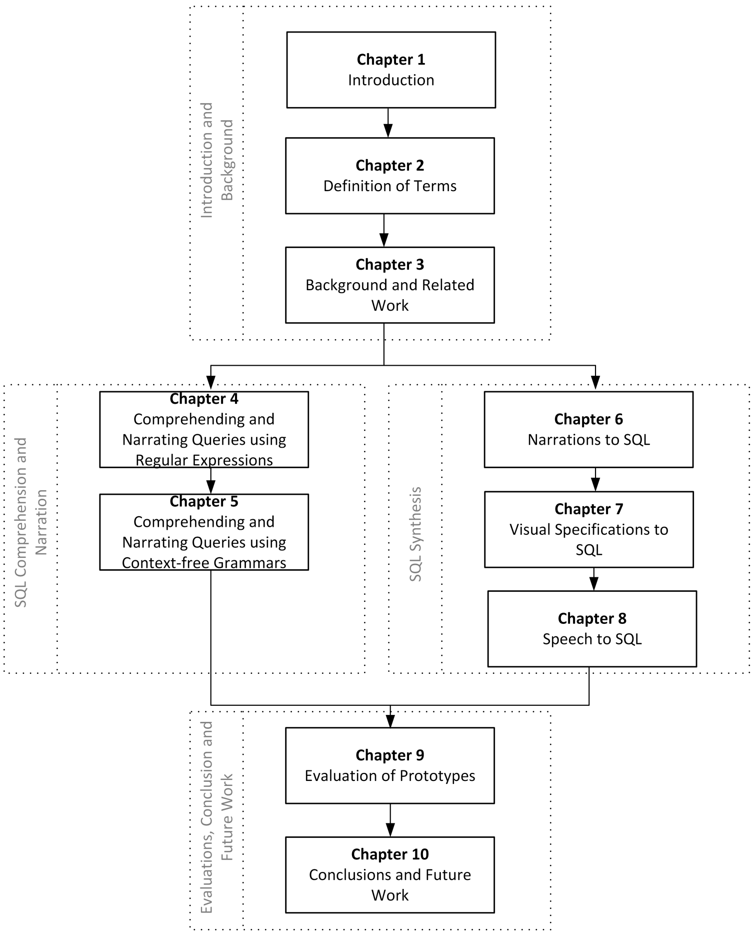


Figure 2: The organisation of this thesis

This thesis is organised into parts with ten chapters as depicted in Figure 2. The description of Parts and Chapters are as follows.

PART I contains two chapters. Chapter 1 presents the introduction of this thesis and the context. Chapter 2 highlights terms and definitions used in this work. Chapter 3 presents the literature reviewed for this work.

PART II focuses on the comprehension aspect using *narrations* for aiding SQL comprehension. In Chapter 4, narrations for simple queries using REs are introduced with detailed illustrations on how this will aid learning SQL queries for the first time. Chapter 5 describes the generation of narrations using CFGs, aimed at assisting a user in understanding nested queries.

PART III describes the synthesis aspect of our work in three chapters. Chapter 6 presents the use of a JFA to synthesise SQL queries from natural language specifications. This aspect shows the use of natural language descriptions to write SQL queries. Chapter 7 introduces a visualiser to generate a query. The visualiser uses drag and drop interactions using visual specifications to generate a query. Chapter 8 describes a speech to query synthesiser, targeted at assisting end-users to write correct SQL queries using speech inputs.

PART IV presents the evaluation and concluding aspects of this thesis in two chapters. Chapter 9 provides the evaluation of the study. Chapter 10 presents the conclusion and future directions of this work.

2 DEFINITION OF TERMS

This chapter presents the definitions of the terms used in this thesis. The terms are listed in different categories across the following areas: FLA, SQL comprehension and concepts, computational linguistics, and other terms.

2.1 FORMAL TERMS

Definition 1 (Lexical Analysis [Grune *et al.* 2012])**.** This is the initial phase of a compiler where program texts are converted into a stream of tokens, white spaces and comments are removed.

Definition 2 (Syntax Analysis [Wilhelm *et al.* 2013])**.** At this phase, the stream of tokens received at the lexical analysis phase is used to produce a tree-like data structure (parse tree or abstract syntax tree). This phase uses a CFG to construct the parse tree. This phase is also called parsing.

Definition 3 (Formal Language Theory [Post 1944; Perrin 2003])**.** The field of formal language theory (FLT) has its root in mathematics. This became popular in 1956 when Noam Chomsky conducted an investigation into natural languages [Chomsky 1956; 1959]. Since its inception, FLT has been applied in different domains such as switching circuits, neural networks, compiler designs (parsers), cryptography and computer graphics [Karhumäki 2007; Kari 2013]. According to the Chomsky [1956] hierarchy, formal languages are categorised into classes of increasing complexity such as regular languages, context-free languages, context-sensitive languages and recursive enumerable languages. Each of these languages is generated from grammars and can be defined by its respective types. For instance, regular languages are defined by regular grammars, etc. Figure 3 presents Chomsky's hierarchy.

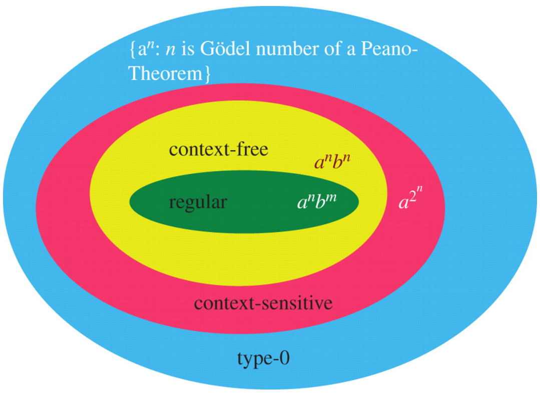


Figure 3: Chomsky's hierarchy of increasing complexity [Jäger and Rogers 2012]

Definition 4 (Basics [Dömösi *et al.* 2016; Chomsky 1956])**.** In this section, we present some basic definitions.

- An *alphabet*, Σ, is a finite set or (collection) of symbols.
- A *string* or (*word*) is a finite sequence of zero or more symbols.
- A *symbol* is an abstract entity or an item.
- A *language*, L, is an infinite collection of strings over some alphabet Σ.

- The symbol Σ^* is a set of finite or (non-empty) strings over Σ, where the symbol (*) is known as the Kleene star.
- The symbol $|w|$ is the length of a string w.
- The symbol λ or ε is an empty string.

Definition 5 (Regular Languages and Regular Expressions (REs) [Ruohonen 2009; Ade-Ibijola 2016a])**.** Let R denote the regular language over an alphabet Σ or R_Σ. The REs that follow are:

1. $\emptyset$ is in R, representing the empty set.
2. λ is in R, representing the empty string.
3. For each symbol a, the language $\{a\}$ in R, representing the regular expression a.
4. Let L_x and L_y is in R, then,
   - $L_x \cup L_y$ is in R, representing the union of both languages.
   - $L_x L_y$ is in R, representing the concatenation of both languages. $L_x{}^*$ is in R, denoting the Kleene star.

Regular languages have a wide range of applications from circuit design and text editing to pattern matching [Câmpeanu *et al.* 2003; Yu 2012]. Their widespread use is due to their highly expressive power. They have become well recognised pattern matching languages [Yu 2012].

Definition 6 (Context-free grammars (CFGs) [Ruohonen 2009; Ade-Ibijola 2016a])**.** CFGs consist of four key components (or *tuples*) such as G = {N, Σ, P, S} where:

1. N is a set of non-terminals (or lexicon) symbols.
2. Σ is a set of terminal symbols.
3. P is a set of production rules.
4. S is a start symbol, where $S \in N$.

Remark. Hence, a language generated by CFG can be repeatedly enumerable by applying production rules P, by starting with the start symbol S, then replacing the non-terminals N with the corresponding production rules P until all non-terminals N have been reached. CFGs can be rewritten in Backus-Naur form (BNF), also denoted as (::=) as presented below, and have been applied to many real-world problems [Javed 2009; Huang *et al.* 2014c]. It is interesting to note that the SQL ISO 2003 uses CFGs for parsing [Schmitz 2007].

Definition 7 (Finite Automaton (FA) or Finite State Machine (FSM) [Kupferman 2018; Genise *et al.* 2019; Meduna and Soukup 2017])**.** A FA or FSM is a 5-tuple where:

1. Q is a finite set of states.
2. Σ is the finite input alphabet.
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function.
4. $s \in Q$ is the start state.
5. $F \subseteq Q$ is the set of final (accept) state.

A FA consists of several parts, which has a set of states and rules for moving from one state to another, depending on the input symbol. FAs are drawn with states as circles, start state indicated by the arrow pointed at it, accept or final states with a double circle and arrows going from one state to another as the transitions. This is shown in Figure 4.

The finite automaton, M, can be described formally by (Q, Σ, δ, X, Z), where:

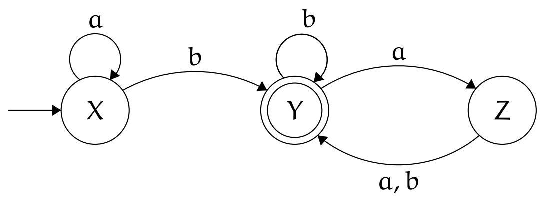


Figure 4: A finite automaton, M, with three states

1. Q = {X,Y,Z} is a finite set of states.

2. Σ = {a,b} is the finite input alphabet.

3. δ : is the transition function, represented by a transition table:

	a	b
X	X	Y
Y	Z	Y
Z	Y	Y

4. $X \in Q$ is the start state.

5. $Z \subseteq Q$ is the set of final (accept) state.

Remark. If the set of all strings (A) that the machine (M) accepts, then A is a language of machine M or $L(M) = A$. A machine may accept several strings but always recognises only one language. If no string is accepted by a machine, M, it still recognises an empty, θ, language.

Definition 8 (A General Jumping Finite Automaton or GJFA [Meduna and Zemek 2012; Křivka and Meduna 2015; Fernau *et al.* 2015]). A GJFA is a quintuple such that M = (Q, Σ, R, s, F) where:

1. Q is a finite set of states.

2. Σ is the finite input alphabet.

3. R is the finite set of rules, where $py \to q$ ($p,q \in Q$, $y \in \Sigma$).

4. $s \in Q$ is the start state.

5. $F \subseteq Q$ is the final state.

If all rules $py \to q \in R$ satisfy $| y | \leqslant 1$, then M is a JFA. Also, L(M) is the language accepted by the automaton. A JFA is based on a Finite Automaton (FA). In a JFA, the input string is not read in a left to right manner. That is, if a symbol M is read, it jumps continuously over a pool of information to an execution point. During computation, a symbol cannot be re-read once it has been read once.

Remark. As a JFA, M is any string in $\Sigma^* Q \Sigma^*$, which represents the binary jumping $\curvearrowright$. It satisfies the condition:

$$vpw \curvearrowright v'qz' \iff \exists\, py \to q \in R\ \exists\, z \in \Sigma^*\colon w = yz \wedge vz = v'z'.$$

It is worth noting that a JFA can be used to represent a Context-sensitive Language (CSL) [Meduna and Zemek 2014; Meduna and Soukup 2017].

2.2 SQL COMPREHENSION TERMS

Definition 9 (Cognitive Workload [Miklody *et al.* 2017])**.** This is the level of measurable effort exerted by a brain in multiple cognitive tasks. Cognitive workload is reflected in the level of brain activity.

Definition 10 (Mental Model [Johnson-Laird 2010; De Boer and Badke-Schaub 2013])**.** This is an abstract notion that builds psychological explanations of how something works. Mental models guide reasoning, behaviour and perception.

Definition 11 (Pedagogical Models [Renaud and Van Biljon 2004])**.** Pedagogical models (or pedagogical patterns) aim to find the best way of teaching for the purpose of sharing knowledge. Patterns have been taken up in many fields and each has a particular structure and vocabulary. For example, the medical science pattern is different from that in software engineering. The pedagogical models consist of:

1. Issues, which involves knowledge transfer of a particular type.
2. Strategy, aimed at transfering knowledge in a particular manner.
3. Implementation, which provides the delivery of content in the way specified by the strategy.

Definition 12 (Theory of Constructivism [Bruner and others 1966])**.** This idea suggests that new ideas can be constructed based upon experiences. The theory shows how humans learn from their past experiences.

Definition 13 (Learning Theories [Pritchard 2017])**.** These are sets of principles that explain how humans acquire, process and retain knowledge. These principles show how learners progress through the phases of learning.

Definition 14 (Learning Taxonomy [Adams 2015; Sarfraz 2017; Verenna *et al.* 2018])**.** This classification shows the skills that educators set for their students to achieve. The taxonomy shows the level of cognition required for a course using a set of objectives. An example of a learning taxonomy is the Bloom's taxonomy of learning. This taxonomy shows the movement from basic (knowledge recall) to highest (evaluation) level of cognition. Other levels fall in between, which include comprehension, application, analysis and synthesis. The different taxonomies of learning discussed in this thesis are Bloom's Taxonomy, Anderson and Krathwol's Taxonomy, Gorman's Taxonomy and the CS Taxonomy.

Definition 15 (Problem-based Learning (PBL) [Hmelo-Silver 2004])**.** This is an instructional approach in which students learn from problem solving. In this method, there is no single correct answer, learners collaborate in groups to identify what is required to solve a problem.

Definition 16 (Pedagogy Style [Renaud and Van Biljon 2004; Mitrovic 2003])**.** This is the principle that shows how a concept should be taught. The SQL pedagogy involves two methods, using instructor-led and electronic tools.

Definition 17 (Syntax-free Approach (SFA) [Pyott and Sanders 1991; Fincher 1999; Ade-Ibijola *et al.* 2014; Ade-Ibijola 2016b])**.** This approach is based on the principle of teaching novices how to program without its inherent syntax. The approach suggested that programming should be taught using clear English terms with the aim of improving program comprehension.

Definition 18 (Narrations [Ade-Ibijola *et al.* 2014; Ade-Ibijola 2016b])**.** Narrations adopted the SFA approach in an attempt to aid program comprehension. This approach uses a high-level descriptions of programs written in plain English often longer than programs they describe. They are also called syntax-free textual algorithms. This approach was employed in this thesis in an attempt to assist users to understand SQL queries.

2.3 SQL CONCEPT TERMS

Definition 19 (Relational Model [Codd 1970])**.** The relational model was developed to model data in the form of relations (tables).

Definition 20 (Relational Database Management System (RDBMS) [Coronel and Morris 2016])**.** RDBMS enables users to create and maintain a relational database. Once the relational database is structured appropriately, it is referred to as *normalisation*. According to Batra [2018], we define the following terminologies to better understand the RDBMS concept:

1. Field: This is the smallest unit of information that consists of a column in a table. A field is also termed an *attribute*.
2. Record: The record consists of each row in a table. This is also regarded as a *tuple*.
3. Table or relation: In the relational model, every relation can be depicted as a table but not every table can be termed as some relation. Hence, a table is a collection of related data organised within a database.
4. Database: This is an organised collection of related tables and data.
5. Query: This is the composition of a table, presented in the form of a predefined SQL query.

Definition 21 (Structured Query Language (SQL) [Hogan 2018])**.** The SQL is a standardised, de facto language for creating and maintaining a RDBMS. Every RDBMS engine supports SQL, which has made it the most comprehensive database language. It consists of statements that support queries, updates and data definitions (DML and DDL). We define these terms as:

1. Data Manipulation Language (DML): The DML consists of commands which are used to load, update and query a database. These consist of statements such as SELECT, INSERT, UPDATE and DELETE.
2. Data Definition Language (DDL): The DDL defines commands for table creation, indexes and views. Popular commands in this category are CREATE, ALTER and DROP.

Definition 22 (Simple Queries [Beaulieu 2009])**.** These are queries that include operations for selection, mapping, built-in functions and simple Boolean values.

Example 2.3.1. An example of a simple query that displays all information from a Student table with a WHERE condition as presented in Listing 1.

Listing 1: A simple query

```
SELECT *
FROM Student
WHERE Name = "Steve";
```

Definition 23 (Nested Queries [Beaulieu 2009])**.** Nested queries include grouping, set operations and relational operators.

Example 2.3.2. A nested query shows nesting properties with multiple SELECT statements as seen in Listing 2.

Listing 2: A nested query

```
SELECT lastname
FROM Student
WHERE lastname IN (SELECT lastname
                   FROM records);
```

Definition 24 (Query Builders [Ceballos *et al.* 2012])**.** Query builders improve the understanding of a query using drag and drop functionality. Typically, a query builder is mostly used to create queries and filters.

Definition 25 (Intelligent Tutoring Systems (ITSs) [Graesser *et al.* 2012])**.** ITSs are learning platforms that incorporate computational models to provide immediate and comprehensive feedback to a learner without requiring an instructor. ITSs evolved from Intelligent Computer-Aided Instruction (ICAI) in 1987 which attempted to produce human-like behaviour [Elsom-Cook 1987]. Such activities are classified as 'good teaching'.

Definition 26 (End-users vs Student [Connolly and Begg 2005])**.** The end-users are the custodians of db applications, who maintain and use data from these applications to serve their information needs. In this work, end-users are classified as:

1. Non-technical end-users. They are typically less knowledgeable of the RDBMS and SQL. The majority of these users are in diverse fields such as marketing, finance, mining, etc. Typically, they access their databases through special-purpose applications to speed up their processing task. For example, QuickBooks[1], a popular accounting software, is mostly used by end-users such as HR managers to carry out payroll tasks.

2. Technical users. Conversely, technical users are familiar with the features offered by the RDBMS. These users are very knowledgeable in SQL, and can write application programs to support their routine tasks. Examples of such users are database administrators, web programmers, data scientists, etc.

A student is an individual who is studying towards a degree. A student can be an undergraduate or a postgraduate. Here, our focus is on undergraduate students as they are studying SQL for the first time. Students and learners are used interchangeably in this thesis.

Remark. In this thesis, our focus is on these users, namely: non-technical end-users in industry, and undergraduate students in academic institutions. These users are our focus because they struggle to write and understand SQL queries.

Definition 27 (Cognitive Models [Al-Shuaily 2013])**.** The cognitive models for SQL involves learning, understanding and remembering. These models require that learners should accumulate a range of contexts so that their formulation and translation skills are improved.

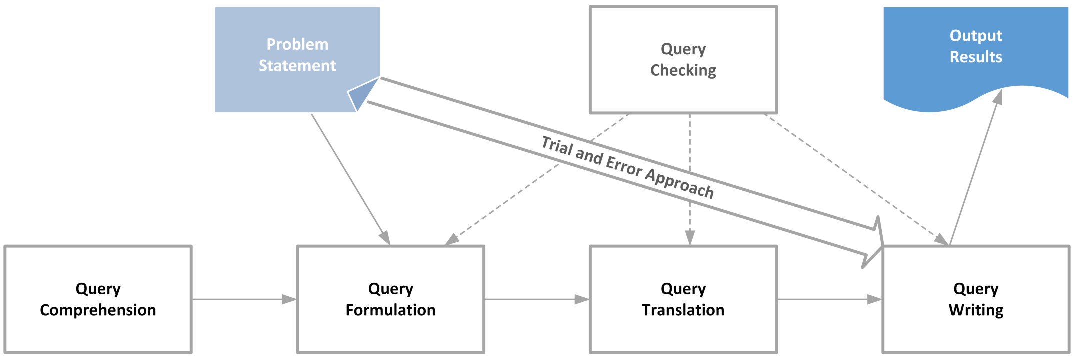


Figure 5: The cognitive models for learning queries [Al-Shuaily 2013]

Figure 5 shows the cognitive models for query learning. The cognitive models are:

1. Query comprehension is the stage that requires learners' skills of reading and understanding of SQL queries. This task involves reading and identifying the required query.

[1] https://quickbooks.intuit.com/

2. Query formulation is an approach that a learner is required to perform during problem solving. This approach requires the student to understand the context of the given problem.

3. Query translation (synthesis) is a stage that requires the learner to express a query in clear English and write the related SQL query.

4. Query writing is a factor that influences how a learner performs once a scenario is given. This phase requires that learners apply their knowledge of the SQL syntax for the provided scenario.

2.4 COMPUTATIONAL LINGUISTICS TERMS

Definition 28 (Natural Language Processing (NLP) [Nadkarni *et al.* 2011; Manning *et al.* 1999]**).** This is a branch of computational linguistics that explores how computers understand words written in human languages. NLP began in the 1950s and has been used in many applications, such as email-spam detection, text summarisation, question-answering (QA), and machine translation (MT), etc.

Generally, NLP is classified into Natural Language Understanding (NLU) and Natural Language Generation (NLG) [Liddy 2001].

Definition 29 (Natural Language Understanding (NLU) [Sharma *et al.* 2019]**).** This concept helps computers to understand and interpret human languages, in speech forms. Most NLUs systems are based on statistical models, which are an important branch of NLP.

Definition 30 (Natural Language Generation (NLG) [Staykova 2014]**).** This process generates a natural language from non-natural language inputs. NLG is also regarded "translator", and classified as a sub-field in Computational Linguistics.

Definition 31 (Natural Language Interfaces to Databases (NLIDBs) [Reinaldha and Widagdo 2014; ElSayed 2015; Sharma *et al.* 2019]**).** NLIDBs are query interfaces, used to translate a natural language query (NLQ) in English into a database query. The first known NLIDB system was LUNAR [Woods 1972], which was developed in the late sixties. Since then, there has been a continuous development of interfaces to assist end-users write queries.

2.5 OTHER TERMS

Definition 32 (Visual Specifications [Rojit *et al.* 2016]**).** Visual specifications are symbols used to represent features, which can be used to display some text or program. In the programming concept, visual specifications have been used to build and demonstrate a programming solution. Examples of visual specification tools are Scratch [Resnick *et al.* 2009], Alice [Dann *et al.* 2011] and Blockly [Fraser 2015]. These tools are generally called Block-based programming languages since they use "drag and drop" interactions to build a program. Figure 6 shows an example of Scratch with annotations describing program blocks.

Definition 33 (Verbal Specifications [Kantorowitz 2014]**).** Verbal specification is a process that clearly expresses the terminology of a domain where the subject matter is less understood. This process involves *speech* forms. Verbal specifications have been applied in many scenarios where a concept is less well understood [Kantorowitz 2014; Meziane *et al.* 2008]. For example, in programming, to assist students produce programs from speech using an informal elicitation to allow users to translate speech into formal Unified Manipulation Language (UML) diagrams [Meziane *et al.* 2008].

Definition 34 (Parsing [Straka and Straková 2017]**).** Parsing is the transformation of a sequence of characters into a syntax tree. During parsing, a sequence of characters such as a sentence, are usually grouped into syntactic parts for recognition. For example,"The dog barks" will be assigned to subject ("The dog") and predicate ("barks").

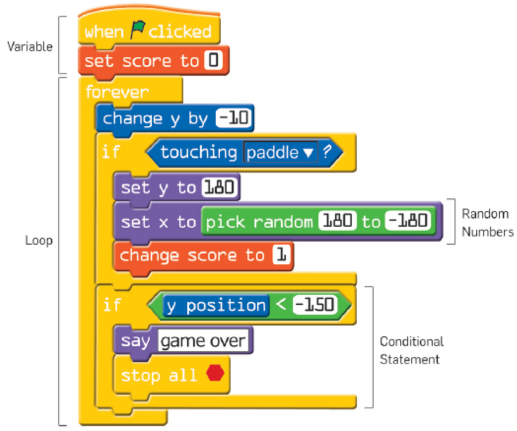


Figure 6: Example of Scratch showing program blocks [Resnick *et al.* 2009]

Definition 35 (Stopwords [Saini and Rakholia 2016])**.** This is a well-established method used to reduce noisy features in textual data. Stopword is based on the idea that words that are not relevant may help produce more accurate results for a classifier. This method is widely used in the NLP domain especially for document classification and information retrieval.

Definition 36 (Stemming [Kotov 2017])**.** This is the process of reducing words to their base form, and has been applied to many computational linguistic problems. For example, the word "going", would be reduced to "go" through stemming. In this case, the gerund (ing) in the word is removed from the base/root form.

2.6 CHAPTER SUMMARY

In this chapter, we have presented the definition of terms used in this thesis. We started by defining terms for formal language and its related languages, then we provided definitions for the SQL comprehension concepts. Next, we defined terms used in the computation linguistic domain, and other newer terms introduced in this work. Chapter 3 reviews the background and related work for this study.

3 BACKGROUND AND RELATED WORK

This chapter reviews the literature related to SQL learning and other related approaches. We start by providing a general overview of SQL and a brief history, then we discuss the challenges associated with learning SQL. Next, we investigate existing learning theories and pedagogies. Furthermore, we highlight state-of-the-art tools used for teaching and learning SQL queries. Last, we discuss recent works in the NLP domain by describing existing methods and related tools.

3.1 INTRODUCTION

It is a well established fact that SQL is predominantly used in the industry and taught as part of an introductory database course in the undergraduate curriculum [Ahadi *et al.* 2015; Bider and Rogers 2016; Grillenberger and Brinda 2012]. For an introductory database course, the basic SQL concepts that students are expected to master are clearly defined [Renaud and Van Biljon 2004; Mitrovic and Ohlsson 2016]. More so, there are many useful resources available that are designed to assist students progress from basic to advanced levels of understanding SQL [Heller 2019c; Bergin *et al.* 2012]. Many publications have presented different pedagogical approaches that address the relevant aspects of SQL that need to be covered by an educator [Renaud and Van Biljon 2004; Lavbič *et al.* 2017; Coronel and Morris 2016; Prior 2014].

Extensive knowledge of SQL skills is vital for many organisations [Liu *et al.* 2003]. Many studies have highlighted that sufficient knowledge of writing correct SQL queries remains a skill that most CS graduates have not mastered [Sander and Wauer 2019; Liu *et al.* 2003; McGill 2008]. A proficiency in SQL is highly sought after by industry employers, and is required for most entry level jobs in programming [Cappel 2002]. A research study by Verma *et al.* [2019] showed that to take up programming roles, an individual must possess knowledge of SQL. Furthermore, Chiang *et al.* [2012] opined that a role in business analysis, data engineering and web development requires extensive knowledge of SQL. Despite the lingering demand for SQL skills, there is still the question of which specific pedagogy and learning style is preferred [Sander and Wauer 2019]. It is a well known fact that teaching and learning are connected processes that help to stimulate a learner [Menekse 2019]. Mayer and Alexander [2016] assert that learning is a process that connects new information to previous knowledge, hence it is imperative to assist learners develop their knowledge of a concept. To support learning, the mode of instruction needs to be designed to facilitate the learning process. Examples are: web-based presentations, educational games or using specialised software programs (or ITSs) [Collins and Halverson 2018]. Despite the usefulness of these instructional materials, students struggle to understand basic and complex SQL queries [Mitrovic 2012; Cembalo *et al.* 2011; Chu *et al.* 2017].

Similarly, non-technical end-users in industry frequently use applications that rely on databases that store structured data [Sagiroglu and Sinanc 2013; Hashem *et al.* 2015; de Silva 2017]. These applications make use of SQL queries to manipulate and retrieve data from these databases. Wang *et al.* [2017a] opined that non-technical end-users would like to access databases and write SQL queries if they had the means to do so. Unfortunately, only technical specialists can accurately write correct SQL queries that extract useful information from these data sources [Elder 2009]. As a result, non-technical end-users must rely on these specialists to generate reports of the query tasks for them. Such an approach can be time-consuming and laborious for an end-user [Zhang and Sun 2013;

Yaghmazadeh *et al.* 2017b; Wang *et al.* 2017a]. A few questions come to mind, including but not limited to: What is SQL? Why is it so hard for students and non-technical end-users to understand SQL queries? What are the main causes of this and how can they be addressed?

This chapter continues by presenting a brief history of SQL in Section 3.2. The challenges encountered when learning SQL are presented in Section 3.3. The pedagogical approaches used for SQL are described in Section 3.4. This is followed by cognitive theories of learning in Section 3.5, and the state-of-the-art tools for SQL are presented in Section 3.6. The NLP methods and similar areas are presented in Section 3.7. The formal language and automata applications are discussed in Section 3.8. The gaps noticed in literature that motivated this research are highlighted in Section 3.9. This chapter concludes with Section 3.10.

3.2 BRIEF HISTORY OF SQL

The idea behind the relational model paved the way for Codd [1970] to release a language called the Structured English QUEry Language (or SEQUEL) to communicate with relational databases. SEQUEL was later renamed to SQL because SEQUEL had already been used for a hardware product [Levene and Loizou 2012; Coronel and Morris 2016]. Since 1986, a joint effort by the ANSI and ISO adopted SQL as the default language for relational databases [Barrera and Pachitariu 2018; Heller 2019b]. Since then, SQL has undergone refinements in 1989, 1992, 1999, 2003, and 2006 [Coronel and Morris 2016]. The current version, SQL:2006, supports the object-oriented functionality, alongside with XML features for querying data, among other updates to the language. Most database vendors such as IBM, Oracle, Microsoft and Informix have continuously used SQL for their products. To date, SQL has remained the preferred language used to communicate with both commercial and open source database products [Harrison 2015].

Many of SQL statements are "English-like", and generally are referred to as a declarative language [de Silva 2017]. The English-like construction indicates that SQL statements are easier to learn and understand. That is, SQL statements resemble English language sentences in their formulation. In addition, SQL is widely considered as a non-procedural language by which its operation is specified (in this context, a result set) rather than a step-by-step computation [Myalapalli and Shiva 2015]. As a result, a database engine decides the tasks and produces a result. Compared to a procedural language such as Java or C, a program can be broken into smaller chunks and a compiler can perform the desired computation in a step-by-step manner. Simply, a user is in control of what the program does. There are two variants of SQL commands: DDL and DML [Dekeyser *et al.* 2007]. The DDL commands allow users to manipulate data in a database, while the DML decides the commands for defining a database schema.

3.3 CHALLENGES OF LEARNING SQL

Despite its simple syntax and highly declarative nature, learning SQL and its underlying concepts pose difficulties for students [Kleiner *et al.* 2013; Jones *et al.* 2016] and non-technical end-users [Li and Jagadish 2016; Soylu *et al.* 2016]. Several authors have tried to identify the reasons for the difficulties encountered by these groups of users while learning SQL. Prior [2003] investigated why students experience difficulties in learning SQL. In this study, students were asked to submit a formal assignment on a piece of paper without having to practise against a relational database. The research highlighted that students perform badly in writing correct queries because most of them are only interested in passing the module rather than taking time to practise sufficiently. Furthermore, the study concluded that constructing and writing correct queries in SQL is a practical skill that cannot be gained without repeated practise. Mitrovic and Ohlsson [2016] opined that learning SQL from a RDBMS often lead to learning challenges. The authors advised that errors generated from most RDBMS are not helpful to learners, because they are limited to the syntax of the RDBMS itself.

Writing DML and DDL expressions have shown to be problematic for learners [Dekeyser *et al.* 2007; Seyed-Abbassi 1993; Qian 2012]. Sadiq *et al.* [2004] presented two reasons for the difficulties experienced while learning SQL. First, the straightforward syntax of the SQL `SELECT` command is often deceptive. To a learner, it might appear easy to learn, but the reverse always seems to be the case. Second, the declarative nature of SQL can be difficult for students to comprehend, especially if they are learning it alongside a procedural or object-oriented programming language. These difficulties were further discussed by Dekeyser *et al.* [2007] and Ahadi *et al.* [2015]. They argued that approaching programming problems in procedural or object-oriented programming languages requires learners to think in *steps*, while SQL requires one to approach a problem in *sets* rather than *steps*.

Another difficulty faced in learning SQL is the burden of memorising database schemas, resulting in inaccurate solutions due to wrong attributes or table names specified [Mitrovic 2003; Mitrovic and Ohlsson 2016; Lavbič *et al.* 2017]. This difficulty often misleads the learner in understanding the underlying concept of SQL. Prior [2014] reported that apart from the burden of memorising database schemas, knowing when they are necessary in writing queries and how to execute them poses great difficulties, requiring consistent practice and effort from learners. Other reasons are that learners misunderstand the use of first-order logic and other basic concepts of SQL [Grust and Scholl 1999; Ahadi *et al.* 2015; Soylu *et al.* 2017].

Kearns *et al.* [1997] suggested that failing to understand basic SQL queries will lead to problems comprehending other concepts such as group and aggregation functions, joins, universal quantification and some set operations. In addition, Ahadi *et al.* [2015] conducted a survey and revealed that a high percentage of students struggle to learn and write correct subqueries (nested and correlated) enclosed in balanced parentheses. The study agreed that only after students have fully grasped the early stages of learning simpler SQL queries, can they begin to understand nested queries as these require a procedural understanding of SQL. Also, recent studies by Cagliero *et al.* [2018] and Taipalus *et al.* [2018] agreed that students should first understand simpler SQL queries before being taught nested queries.

Similarly, non-technical end-users struggle to understand SQL queries written by technical experts [Ardito *et al.* 2014; Li and Jagadish 2014b]. If a database administrator leaves an organisation, and the non-technical user is left to use an enterprise application without proper support and training, this can result in redundant reporting, which contains repetitive and unwanted data that makes it difficult to use in these environments, since the users do not understand queries. In most cases, enterprise applications that use SQL as a back-end are often employed by these users, which are often provided in a software documentation [Warnke 2009]. These materials may contain terminologies that are difficult to understand and interpret. If the documentation is not properly designed with non-technical end-users in mind, this may pose serious difficulties.

The majority of these non-technical end-users work in diverse fields such as marketing, finance, mining, etc. Many of these users are faced with the challenge of retrieving vital information from their databases. In most cases, they can clearly specify the intended task, but lack the knowledge to write a correct SQL query. Thus, end-users often seek help from technical users or through online forums [Wang *et al.* 2017a;b]. Such a process can be time-consuming and frustrating [Yaghmazadeh *et al.* 2017a]. In view of these challenges, we discuss the pedagogical patterns and learning strategies that have been proposed over the years.

3.4 PEDAGOGICAL MODELS

Pedagogical models (or pedagogical patterns) and their use in teaching have been widely researched and debated over the years [Kotzé *et al.* 2008; Schulte *et al.* 2010]. Instructors apply different pedagogical models in their teaching curriculum. The father of patterns, Alexander [1977], defined patterns as "each pattern described a problem that occurs repeatedly, used to describe the solution to a problem over a million times". This quote suggests that patterns give designers the freedom of problem solving through many variations.

Bergin *et al.* [2012] describe pedagogical models as the detailed description of work carried out by an educator to communicate knowledge to others and to solve recurrent problems. Renaud and Van Biljon [2004] added that pedagogical models consist of three forms, namely: issues, strategies and implementation. The issues refer to the transfer of knowledge. The strategy aims to transfer knowledge in a particular way and implementation describes the materials used by the strategy. In a teaching and learning scenario, patterns offer a way of transferring knowledge. The intent is to capture and present a concept in a compact form to those that require the knowledge. We present the different pedagogical patterns as discussed in the literature.

3.4.1 *Models in Human-Computer Interaction*

The first human-computer interaction (HCI) patterns were based on user-centered system design with reference to Alexander's ideas [Borchers 2000]. These patterns were discussed in the Common Ground [Tidwell 1999], Designing Interfaces [Tidwell 2010] and User Interface (UI) patterns and techniques [Tidwell 2002]. HCI pattern was defined by Dearden and Finlay [2006] as a description of a proven solution for a user interface that takes place within a particular context. HCI patterns are also referred to as UI patterns that assist software developers to reuse best practices and avoid reinventing the wheel [Seffah 2015]. The study showed that patterns are applicable to every software system and are widely independent of the tools that are used to develop those systems.

Borchers and Thomas [2001] discussed that HCI patterns should model design experience based on the field of architecture where pattern ideas were orginally conceived. The authors stressed that HCI patterns should focus exclusively on the non-technical end-user to embrace their potentials. Using patterns only for an expert user would limit their importance. Van Welie and Van der Veer [2003] proposed a top-down approach of HCI patterns organisation into a scale of hierarchy from high-level design problems which are gradually unpacked into low-level design problems.

3.4.2 *Models in Education*

Just as patterns are employed in many fields to teach students about certain concepts, their application in education are numerous [Borrego 2007; Vermunt and Donche 2017; Hansen and Reich 2015]. Al-Shuaily [2013] described patterns in education as a technique to help educators transfer their experience in a manner that helps to achieve good teaching and learning. In this thesis, the author suggested that using patterns in a CS class will enhance problem solving skills. Educational patterns describe successful practices within an educational context that includes methods, content and curriculum design [Winthrop and McGivney 2015].

Laurillard [2013] argued that the current teaching approach is changing and teachers are required to cope with a technological environment. In addition, the author stressed that teaching should be seen as a creative design profession. Clayer *et al.* [2013] proposed an engineering framework aimed at collecting different pedagogical designs from instructors. The result showed that the tool allowed instructors to convey their opinion in a self-expressive manner.

3.4.3 *Models in SQL*

In teaching SQL queries, pedagogical patterns employed either make use of traditional face-to-face instructor-led teaching [Mao and Brown 2007] or electronic aids [De Raadt *et al.* 2007; Dollinger 2010]. Prior and Lister [2004] proposed three approaches to teaching SQL. The first aspect of this approach required that students be graded using a method that would help improve their learning skills. The idea was that giving students practicals and grading them would improve their query formulation skills. The second part of this approach considered improving SQL skills by using real-world software development, while the third process encourages students to build their SQL query skills by practising online. A similar study was conducted by Abelló *et al.* [2008], who encouraged the automatic grading of students which would enhance and improve their SQL skills.

Another pedagogical pattern for teaching SQL was presented by Renaud and Van Biljon [2004]. The authors compared and contrasted two approaches to teaching in terms of mental models and cognition. The first approach exposed students to using tools to learn the SQL syntaxes, while the second approach required students to formulate queries on paper for weeks before they are exposed to tools. The results showed that if learners are taught using a particular paradigm, exposure to other methods would compromise their SQL query formulation skills. They insist that students need to grasp the basic concept of SQL before they are exposed to tools. Caldeira [2008] conducted a similar study, which required that students understand SQL thoroughly, by reading and understanding how to write SQL scripts, before they are exposed to tools.

Ahadi *et al.* [2016] presented common semantics that instructors need to consider when teaching students to write SQL queries. They emphasised that a deeper understanding of these semantics would improve students' learning outcomes and proper writing of SQL queries.

3.5 LEARNING APPROACHES

The study of how humans learn is not new. As the study of learning continues to expand, researchers have continually applied their ideas to this concept. Over the past few decades, learning theorists have engaged in extensive debates on how people learn [Mowrer 1960; Gredler 1992; Seligman 1970]. Similarly, these theorists agreed that there is no clear definition of learning that is universally accepted. We provide some of the definitions in the literature. Lachman [1997] refers to learning as a behavioural change due to experience. De Boer and Badke-Schaub [2013] defined learning as an activity that involves acquiring and modifying knowledge, attitude, skills and behaviours.

Schunk [2012] identified three criteria for learning. First, learning involves change – people learn by doing things differently; second, learning endures over time – those changes are temporary and may not last forever; and, learning occurs through experience – humans learn from past experience. De Houwer *et al.* [2013] argued that not all of these definitions are valid; a change in behaviour may not necessary imply learning. Furthermore, the study concluded that a change in behaviour is only caused by some experience in an individual and may not count as instances of learning. Eberl and Kaiser [2018] retorted that some definitions need to specify that learning requires changes in a specific psychological mechanism to clearly make a distinction between a behavioural change and learning.

The act of learning is an active area of research in psychology, behavioural ecology, neuroscience, CS and many other disciplines [Barron *et al.* 2015]. In the CSs, students learn how to program in the early stages of their academic pursuit [Resnick 2013; Moreno-León and Robles 2016; Hu and Shepherd 2013]. Learning how to code is considered a hot skill and most industry employers are in dire need of software programmers [Luxton-Reilly *et al.* 2018]. At the early stages, most students are introduced to tools to enable them to improve and learn programming [Moreno-León and Robles 2016]. Even so, students are taught SQL queries in introductory database courses, but learning how to write correct queries is problematic [Ahadi *et al.* 2015; Soylu *et al.* 2017]. It is important to know what modes of learning exist in order to ensure knowledge impacted by the instructor through instructional materials, is understood and learned. The next section describes the different learning strategies.

3.5.1 *Learning Theories*

Learning theories are conceptual frameworks that explain how humans acquire, retain and recall information [Politis 2008]. An effective instructional design is important to ensure clarity, direction and focus throughout the learning process [McLeod 2003]. Schunk [2012] noted that learners progress through different learning stages from novice to expert, and most learning theories share common instructional principles which aid the learner. That is, learning theories assist instructors in designing instructional contents to facilitate learning. Many educational psychologists have identified different learning theories that explain how individuals learn by acquiring and organising knowledge [Politis 2008; Ertmer and Newby 2013; Hung 2001]. According to Hung [2001], the four learning theories are behaviourism, cognitivism, constructivism and social constructivism.

3.5.1.1 *Behaviourism*

Behaviourism was the first learning theory developed in the late 1800s and early 1900s [James 2006]. According to behaviourism, the goal is to derive an elementary law of learning and behaviour that can be extended to more complex scenarios. Davey [2017] described the behaviourist theory as a theory based on all behaviours which are acquired through conditioning. Behaviourists believe that conditioning occurs through the environment and our responses to environmental factors shape our actions [Sheldon 2011; Done and Murphy 2018]. Merriam *et al.* [2006] highlighted three assumptions of learning used in the behaviourist theory:

1. All behavioural-related tasks have little regard for a learner's cognition.
2. Learning activity is influenced by environmental factors.
3. Events formation and reinforcement form important components of the learning process.

These assumptions imply that learning is based on time-controlled events and environmental factors which bring about a change in behavioural responses. The strengths and weaknesses of the behaviourist theory were identified by Engeström and Sannino [2012] and Omar [2018]. In their works, the main strengths are that an individual is expected to behave in a certain way regardless of the circumstance. The weakness of this theory is that the mental cues that a learner receives may not match what was previously learnt.

3.5.1.2 *Cognitivism*

Cognitivism is based on the concept that learning is more important than responses to external factors such as the environment [Duffy and Jonassen 2013; Harasim 2017]. Cognitivism suggests that a reorganisation of experiences could lead to learners making sense of what they are taught, which can be termed learning [Mayer 2009]. Tobin [2012] argued that each learner experiences things by generating their own rules and mental models. Hence, the adjustment of mental models paves the way for newer experiences. Kalina and Powell [2009] noted that before instructors can start designing instructional materials, they need to consider the student's learning point and allow them to create personal meaning before knowledge can be passed to them. Modern instructional methods are based on cognitive theories [Harasim 2017; Kalina and Powell 2009].

The merit of cognitive theories is that learners are trained so that they are able to accomplish tasks on their own [Omar 2018]. The weakness of this theory as emphasised by Omar [2018] is that learners are forced to learn and accomplish tasks in a certain way. For example, in programming, learners can produce different working versions of a program, but some may be more efficient than others.

3.5.1.3 *Constructivism*

Pritchard [2017] described constructivism as the idea that learners can construct knowledge for themselves. Here, each learner constructs meaning as they learn, which is a vital part of the learning process. Constructivism was formalised by Jean Piaget, who suggested that through assimilation, knowledge can be constructed from experience [Kalina and Powell 2009]. Since constructivism describes how learning happens, it is not a particular pedagogy [Hein 1999]. It suggests how learners can use their previous experiences to understand instructional materials. Constructivist theory also places emphasis on mental processes of the learner. It is required that a learner utilises different cognitive processes for tasks. Hein [1999] listed some underlying principles that guide constructivism theory:

1. Learning is an active process that learners can use to construct meaning.
2. Learning involves constructing meaning: people learn to learn as they learn.
3. The action of constructing meaning is mental.
4. Learning is a social activity.
5. Motivation is essential to learning.

A major advantage of this theory is that real-world situations are understood by relating them to past events [Omar 2018]. Where conformity in thinking and actions are required, this learning theory may not be the best approach [Omar 2018].

3.5.1.4 *Social Constructivism*

The theory of social constructivism was developed by Lev Vygotsky in the 1930s and was the first one to reject the claim made by Piaget [Hirtle 1996; Pass 2007]. Chaiklin [2003] interpreted Vygotsky's work to indicate that learning can be separated from its social interactions. Powell and Wimmer [2015] described the social constructivism theory of learning as an effective method that involves collaboration and social interaction. This type of learning is based on social interactions that are developed alongside personal critical thinking in the classroom. The study concluded that adding this form of learning alongside the constructivist approach would improve active participation in a classroom.

Social constructivists encourage learners to develop their own version of knowledge by learning from knowledgeable experts through social interactions [Hein 1999]. This study also adds that knowledge is distributed in the community and can be built through engagements. The social constructivist approach assists young children to develop their knowledge by interacting with their immediate peers, adults and the physical world [Hurst *et al.* 2013]. Hence, from this viewpoint, learning is acquired and improved as the immediate community helps shape the knowledge.

3.5.2 *Learning Taxonomies*

Learning taxonomies are used to describe different aspects of learning behaviours in the form of a classification [Horn *et al.* 2017]. This classification shows the skills that educators set for their students to achieve [Sarfraz 2017]. The taxonomy shows the level of cognition required for a course using some set of objectives [Verenna *et al.* 2018]. Learning taxonomies usually move from basic to higher levels of cognition. This section presents some different learning taxonomies.

3.5.2.1 *Bloom's Taxonomy*

Bloom's taxonomy was devised in the 1950s, and was regarded as the stairway of learning that instructors use to enable students to reach a higher cognitive level [Krathwohl 2002; Adams 2015]. Since then, Bloom's taxonomy has stood the test of time [Sarfraz 2017]. The taxonomy divides the cognitive aspects of learning into six hierarchical levels with increasing complexity [Starr *et al.* 2008]. The hierarchy ranges from the highest (analysis, synthesis and evaluation) to the lowest levels (knowledge, comprehension and application). Each cognitive level of Bloom's taxonomy assists the next level. The taxonomy can be used in almost all disciplines [Sarfraz 2017]. Figure 7 shows the Bloom's taxonomy of learning.

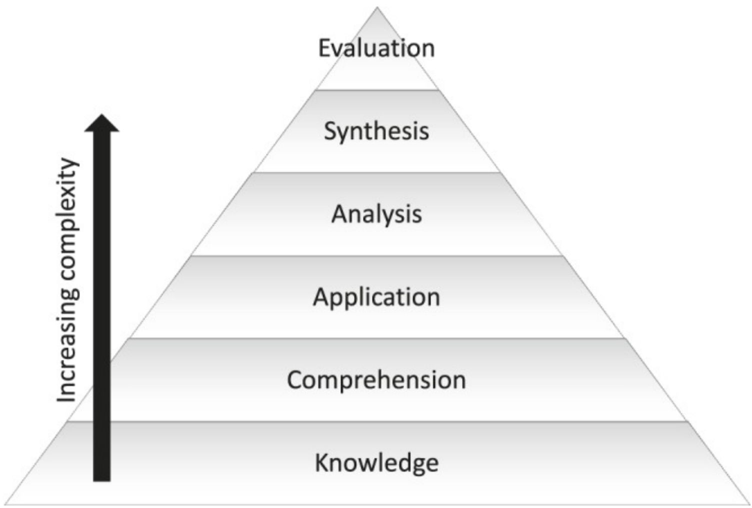


Figure 7: Bloom's taxonomy of learning [Adams 2015]

Crowe *et al.* [2008] applied Bloom's taxonomy in a biology study to assist instructors in creating instructional materials and successfully design questions that require students to apply their cognitive skills. The study showed that Bloom's taxonomy could help students to become successful biologists as this strategy can improve their learning skills. Thompson *et al.* [2008], in their study, identified that many learning theorists believe that it was difficult to apply Bloom's taxonomy in introductory programming courses. In addition, the study reiterated that Bloom's taxonomy can be a very useful tool for the CSs discipline because it addresses cognitive processes that programmming courses require. Hence, it can be an invaluable tool for CS educators.

3.5.2.2 *Anderson and Krathwohl Taxonomy*

Bloom's taxonomy was revised to allow educators to understand and implement standards in their curriculum [Krathwohl and Anderson 2001; Forehand 2010]. This revised taxonomy is regarded as the Krathwohl and Anderson [2001] taxonomy. The taxonomy maps six well organised cognitive processes into knowledge levels and takes into consideration many of the criticisms levelled against Bloom's taxonomy.

In comparison with Bloom's taxonomy, the Anderson and Krathwohl taxonomy is more comprehensive and has shown to assist instructors in designing instructional materials [Pickard and others 2007]. In its formation, it reword nouns used in Bloom's taxonomy as verbs. In this taxonomy, the 'synthesis' phase of the Bloom's taxonomy was replaced with the 'creating' phase at the top of the pyramid Thompson *et al.* [2008]. Similarly, the lowest level of the Bloom's taxonomy called 'knowledge' was replaced with 'remembering'. Although Bloom's taxonomy is focused on the learning process in many forms, a disadvantage thereof is that it fails to indicate that learners must start at a lowest level before working up [Churches 2010]. Instead, the learning process can be initiated at any point (context-free), and the learner can perform the learning task alongside. Figure 8 presents the Anderson and Krathwohl's revised taxonomy.

Jansen *et al.* [2009] suggested that although the Anderson and Krawthwohl taxonomy was effective in investigating the cognitive learning processes of learners, developing questions for instructors is not straightforward.

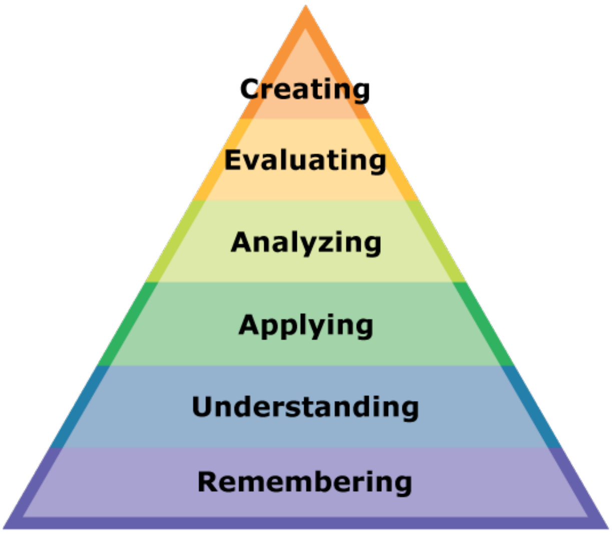


Figure 8: The Anderson and Krathwohl's revised taxonomy [Wilson 2018]

3.5.2.3 *Gorman Taxonomy*

Michael Gorman proposed a taxonomy that consists of four simple learning levels [Gorman 2002]. These levels shows how knowledge can be represented, whether implicitly or explicitly. Figure 9 includes the lower levels information (what) and skills (how) with higher levels of judgement (when) and wisdom (why). Al-Shuaily and Renaud [2010] compared Bloom's and Gorman's learning taxonomies. The study indicated that the lowest level, 'what' aligned with 'knowledge and comprehension' in Bloom's taxonomy. In addition, the higher level word 'why' matches with 'evaluation.

Mohtashami and Scher [2000] noted that teaching the database concept with Gorman's strategy would require the first level to cover basic aspects such as entities, relations, etc. The second level should incorporate concepts such as ERDs[1] and SQL. The final level should

1 Entity-Relationship Diagrams

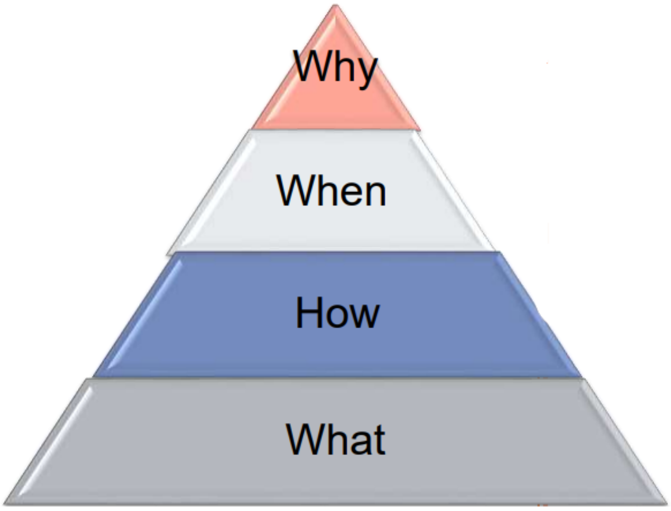


Figure 9: Gorman's taxonomy of learning [Gorman 2002]

investigate a problem-based approach to enable students use previous knowledge to tackle questions for the purpose of understanding the database concept.

3.5.2.4 *Computer Science Taxonomy*

Many educators have applied different taxonomies to the CS discipline [Maier and Größler 2000; Johnson and Fuller 2006; Rutten *et al.* 2012]. Johnson and Fuller [2006] carried out a survey using Bloom's taxonomy to examine whether it is appropriate for the CS field. The survey showed that Bloom taxonomy can be helpful for educators developing instructional materials for their courses. In a similar study, Lahtinen [2007] investigated Bloom's six cognitive activities for its use in CS education. The study revealed that the taxonomy was indeed useful to CS educators. Bower [2008] presented a learning taxonomy that identifies different programming processes undertaken by students when learning programming. The taxonomy showed that students are encouraged to focus on tasks that foster memory retention.

Shneiderman [1978] introduced five learning tasks required for SQL learning. These tasks showed that learning SQL would require a learner to understand the syntax and semantics first before modifying queries written by oneself. Renaud and H.A.S. [2009] argued that before learning SQL, it is important that query construction skills are developed first. The study recommended Gorman's taxonomy, where students are required to learn the basic aspects first, before moving to a higher level.

3.5.3 *Learning Styles*

As is widely discussed in the literature [Entwistle and Ramsden 2015; Lowe *et al.* 2016; Adams 2017], individuals have unique ways of learning and processing information. The benefit derived from learning content and materials that match an individual's learning styles has been identified in the works of Price [2004] and Fleming *et al.* [2011]. This has also been identified in some computer-assisted learning systems [Truong 2016; Sweta and Lal 2016]. The terms 'cognitive styles' and 'learning styles' are used interchangeably in the literature. Cassidy [2004] defines an individual's cognitive style as a problem-solving, thinking, perceiving and remembering activity, while learning style indicates the application of cognitive style in a learning situation. Soflano *et al.* [2015] defined learning style as an individual's choice and strategy for achieving learning objectives efficiently. In the domain of SQL, different learning approaches have been proposed. We discuss the different learning methods, such as PBL, learning from worked examples, learning through visualisation and learning from errors.

3.5.3.1 *Problem-based Learning*

PBL is a constructivist[2] approach in which learners learn through problem solving [Connolly and Begg 2006]. In PBL, learning is based on acquiring and processing information that changes the knowledge previously acquired by a learner. In PBL, learning is achieved using problems to motivate students and there is a focus on student-centred activities. Using PBL, instructors act only as facilitators rather than as primary sources of knowledge [Hoic-Bozic *et al.* 2009]. PBL is often applied to improve results in collaborative learning where students work in small groups [Thurley and Dennick 2008; Azer *et al.* 2013; Chao 2016]. In collaborative learning, the task of finding the solutions to problems is shared among the group. Figure 10 describes a game-based problem task for a user, where the user has to solve a series of problems before accomplishing a specific task.

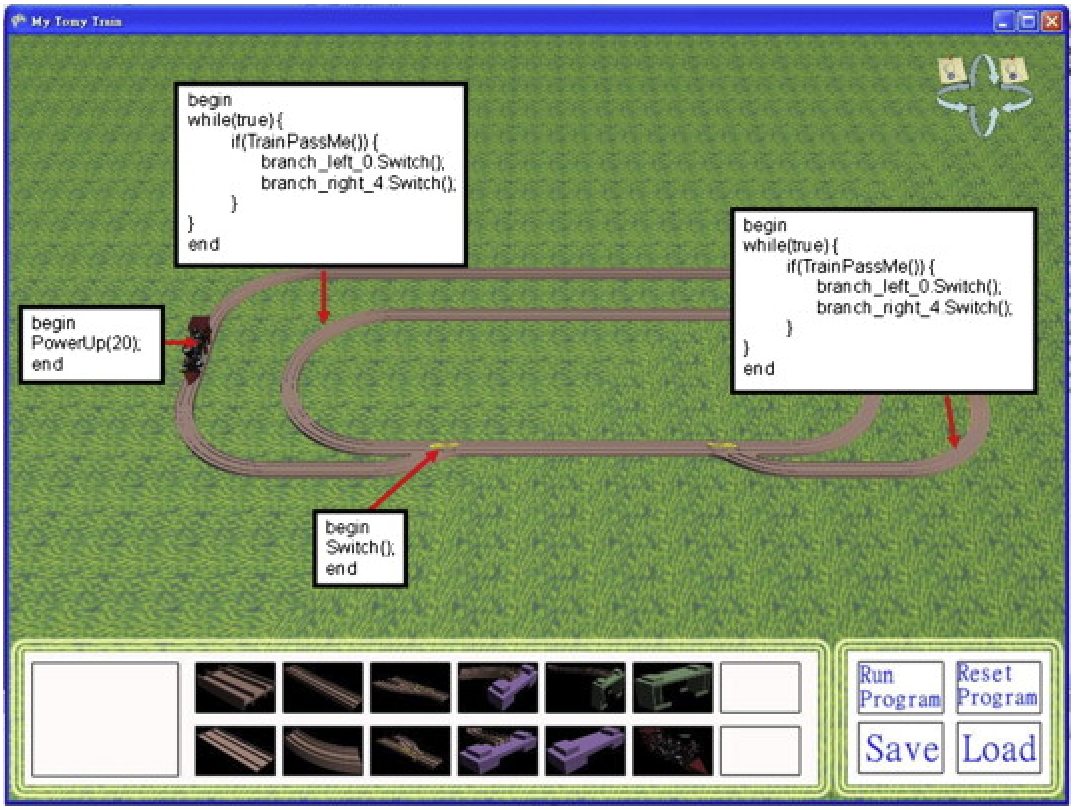


Figure 10: Problem-based task in a gaming scenario domain [Liu *et al.* 2011]

Kreie and Ernst [2013] explored the use of the PBL approach to improve database learning. In their study, students were asked to create a logical data model prototype of a database application. The prototype would require them to solve some data modelling errors which they would encounter. The researchers concluded that the PBL approach would challenge students to understand database concepts and improve their knowledge. Ward [2015] extended the use of the PBL approach in teaching SQL through a game. The research presented a number of problem-solving skills for a learner in a typical gaming scenario. The researcher concluded

2 Constructivist - learning based on observation or through a scientific approach

that using the PBL approach, a learner would have a deeper interaction in solving problems in SQL.

3.5.3.2 *Learning from Worked Examples*

According to Sweller [2006], learning from worked examples is the most effective learning strategy, especially when first learning a new domain. Worked examples are presented to students, followed by problem-solving techniques. This is done to ensure that they acquire adequate knowledge before being introduced to problems. Such an approach is ideal for novices, since examples help reduce cognitive workload and aid initial stage learning [Van Gog *et al.* 2011; Renkl 2014]. Najar *et al.* [2014] proposed the use of learning from worked examples ITS for teaching SQL. The study concluded that students' understanding increases significantly when presented with worked examples. Figure 11 shows an example of a probability worked example in mathematics, with the colours indicating how the answer is derived.

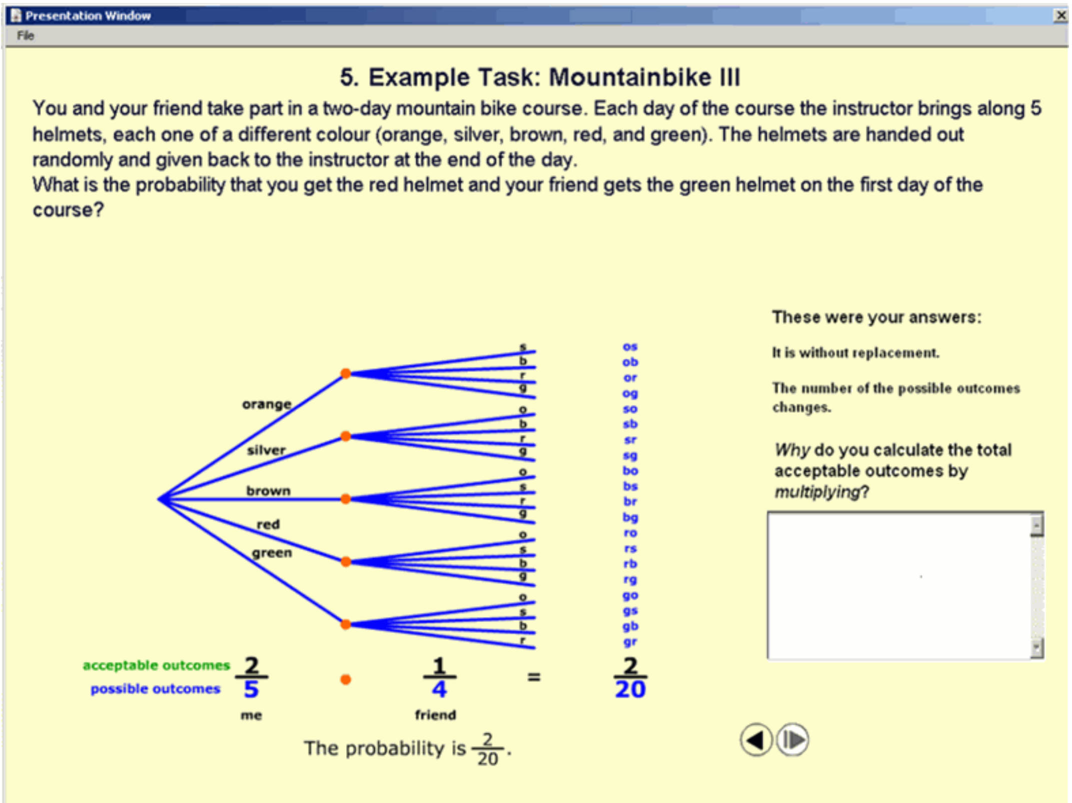


Figure 11: A worked example in the mathematics domain [Berthold *et al.* 2009]

Another study of learning from worked examples was presented by Chen *et al.* [2017a]. The study showed that in most cases, worked examples contain the right explanations for each and every step required. This helps novices gain the right information through the examples provided. The researchers concluded that learning from worked examples is ideal for novices rather than for advanced students.

3.5.3.3 *Learning through Visualisation*

Learning through visualisation has proved useful in assisting novices understand a concept [Watson *et al.* 2011; Garner 2003; Kinchin 2011]. This technique has been used extensively in different application domains to present ideas [Ellis and Mansmann 2010; Keim *et al.* 2008]. Ellis and Dix [2007] defined visualisation as a systematic way of representing an abstract idea that facilitates human understanding. This abstract idea is usually designed in a way that is "playful and aesthetically pleasing", so that users can explore how to solve tasks. Visualisation can encourage active participation in learning and lead students' critical thought processes [Allenstein *et al.* 2008].

A study conducted by Kellems *et al.* [2016] showed that visualisation can even help students with learning disabilities grasp information more easily. Their study showed that visual aids can also better meet the academic demands of students with autism spectrum disorder (ASD)[3], since they do not require intensive training. A sample visualisation tool is presented in Figure 12. This example shows how query statements are interconnected.

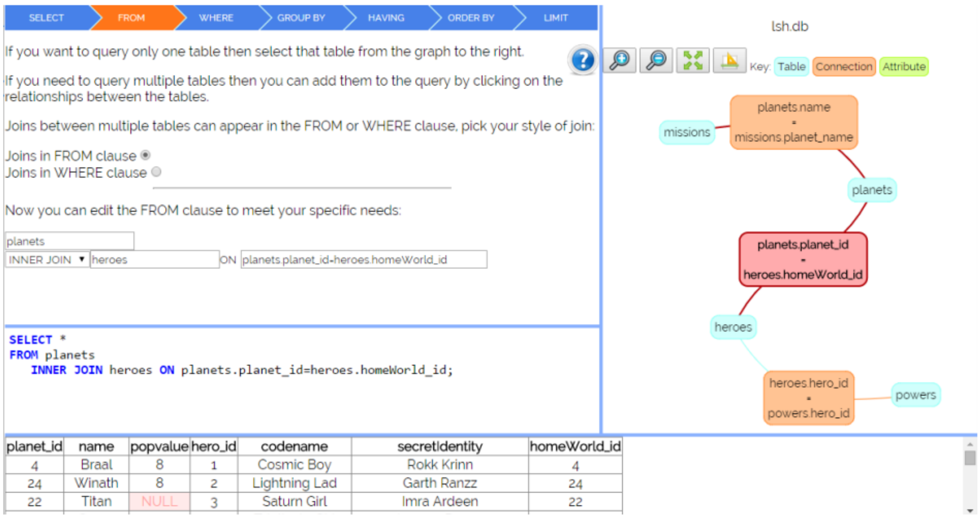


Figure 12: Sis, a visualisation tool [Garner and Mariani 2015]

In the area of program comprehension, visualisation has been explored to aid the understanding of programming [Yassine *et al.* 2017; Kinchin 2011]. Lee *et al.* [2013] proposed the use of the 'drag and drop' refactoring visualisation to assist programmers comprehend programs written in Java. Empirical evidence presented in their work showed that the approach was more efficient and less error-prone, and that it could help programmers comprehend programs easily. Studies conducted in block programming (a technique which represents programs as blocks and uses the drag and drop technique to generate a program) indicated that this type of visualisation increased student engagement and that it was effective in knowledge transfer [Malan and Leitner 2007; Rizvi *et al.* 2011].

A recent study with similar methodology, conducted on serious games, found that this approach paves the way for undergraduate students to learn programming [Yassine *et al.* 2017]. The benefit of this visual aid is that it enhances self-pacing, while its entertaining component attracts the attention of students and engages them in the learning process. It is worth noting that most SQL comprehension tools employ visualisation to comprehend SQL queries and database schemas automatically either through textual or graphical representations [Sadiq *et al.* 2004; Cembalo *et al.* 2011; Folland 2016].

3.5.3.4 *Learning from Errors*

The concept 'learning from errors' or LFE has been successfully applied in mathematics [Brodie 2014], physics [Große and Renkl 2007] and CS (especially in programming) [Shah

3 A neurological and developmental disorder, which result in communication and interaction difficulties.

et al. 2017]. A recent study by Metcalfe [2017] showed that errorful learning followed by constructive feedback is a vehicle to achieving purposeful learning. Other research conducted by Metcalfe and Xu [2017] showed that learning from one's own errors and those of others is useful in helping to develop methods to improve students' learning.

Mitrovic [2012] applied the use of constraint-based modeling in SQL in dealing with errors. The study identified that fixing errors is a time-consuming process that requires a great deal of mental effort. Most importantly, people make errors because their procedural knowledge is poor. The study concluded that if one wants to learn from errors in SQL, one must first accumulate adequate declarative knowledge, which is later converted to procedural knowledge, a process requiring much practice. Katz and Shmallo [2016] extended LFE to the area of relational database modelling to examine the difficulties faced by students in understanding conceptual database modelling. The study showed that errors play a powerful role in database modelling, which encourages students to possess a deeper understanding of the course.

3.5.3.5 *Other Learning Styles*

While we have discussed only a few learning methods, other learning theories have been proposed that are used in computer-assisted learning. The learning theories, as discussed in the work of Soflano *et al.* [2015] are:

1. Kolb's learning style, a model developed by Wolfe and Kolb [1984] which is based on four elements: reflexive observation, concrete evidence, abstract conceptualisation and active experimentation.

2. The VAK model, a model proposed by Dunn and Dunn [1978], divides students into groups based on their learning preferences: visual, auditory or tactile.

3. The Big-5 model, an approach by Felicia and Pitt [2009], has five elements: openness, conscientiousness, extroversion, neuroticism and agreeableness.

4. Honey and Mumford's model, proposed by Honey *et al.* [1992], has four elements: activists, reflectors, theorists and pragmatists.

5. The Felder-Silverman learning model, developed by Felder *et al.* [1988], which consists of four elements: perception, input, processing and organisation.

3.5.4 *Cognitive and Mental Models*

Cognitive strategies that influence the effectiveness of teaching and learning are useful in education [McDougle *et al.* 2016; Lane 2012]. Hence, cognitive models are representations of how humans gain knowledge. Gentner and Stevens [2014] described mental models as concepts used to provide explanations for the purpose of understanding. These concepts may be from artifacts of technology, environmental factors, or tasks that need to be understood. They are mainly explanations that describe how novices understand a concept. This section provides some cognitive and mental models that have been used to support the understanding of SQL, as well as a number of cognitive theories discussed in the program comprehension domain.

3.5.4.1 *SQL Models*

Many studies have investigated different cognitive theories used to support the learning of SQL [Hatami 2017; Schlager and Ogden 1986; Al Shuaily and Renaud 2016a]. SQL learning cognition, as defined by Robins *et al.* [2003], is the construction of schemas, which are organised chunks of related knowledge. The study further shows that learning either builds new schema or modifies existing schemas. Hence, to build cognitive processing of SQL, a mental model must be built to modify or construct knowledge in a schema.

Al-Shuaily [2013] discussed that a complete understanding of SQL requires the novice to draw up mental models of the syntax and semantic concepts. The author noted that these concepts must be understood and their usage in a given scenario is very important to build good mental models of SQL. In addition, the study described four cognitive models for SQL to improve learning, understanding and remembering. These models are query comprehension, formulation, translation and writing to accumulate a range of contexts so that learning can be improved. In an extended study, Al Shuaily and Renaud [2016a] argued that the schemata approach may not be the preferred solution. The study advised that knowledge of syntax and semantics is not sufficient to achieve mastery of SQL queries. The study added that a trial and error approach of solving schema problems may offer better cognitive processes for novices to achieve mastery of query problems. Figure 13 illustrates both approaches used to show mental processes and how problem solving is constructed.

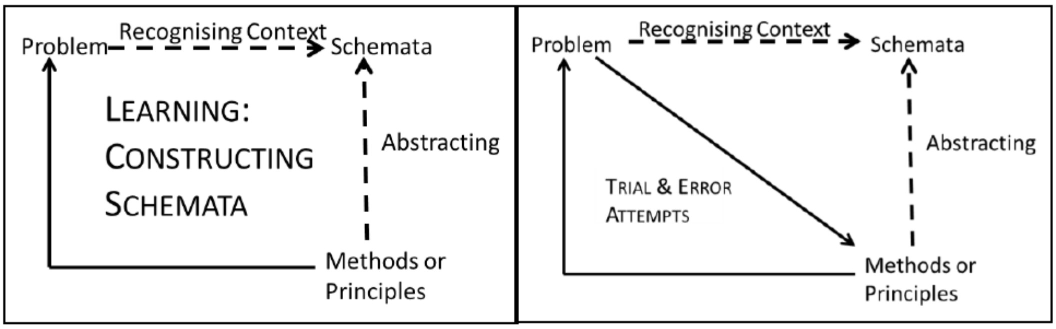


Figure 13: The schemata, and trial and error approaches [Al-Shuaily 2013; Al Shuaily and Renaud 2016a]

Mason *et al.* [2016] proposed a cognitive load theory (CLT) that addresses poor performances in an introductory database course, especially in the SQL concept. The study includes the human CLT model proposed by Sweller [1988] that focuses on long-term and working memory of knowledge storage as schema. The schema used in the study indicates cognition rather than database structure description. The study showed that schemas are retrieved from long-term memory and once they become complex, more information is used to expand the working memory. The human CLT concept is described in Figure 14.

In recent studies, Reisner [1977] described a study that generates and merges a set of lexical items and query templates for query generation. The study was followed by Mannino [2001], who proposed a two step approach that generates a query from problems and database representations. In addition, other studies expanded on the work of Mannino [2001] by discussing three cognitive approaches to the SQL query generation [Siau and Tan 2006; Ogden 1986]. The approaches are query formulation, query translation and query writing. The study concluded that combining these approaches will aid students in solving SQL problems. These approaches are summarised in Figure 15.

3.5.4.2 *Programming Models*

We present some of the programming cognitive models, as we take a cue from the SFA introduced by Fincher [1999]. This approach was used in the current research. Program cognitive models were described as the representation of a program in a developer's memory [Lemut *et al.* 2013; Gulwani *et al.* 2015; Ade-Ibijola 2016b; Storey 2005]. These theories were used to form a mental model useful to ensure that programs should be well understood. These models are described in this section.

Schulte *et al.* [2010] presented the *top-down* approach as an assimilation process that applies knowledge of the program domain and maps this to the code itself. At this stage, the assimilation process is driven by hypothesis and used to convey some information about

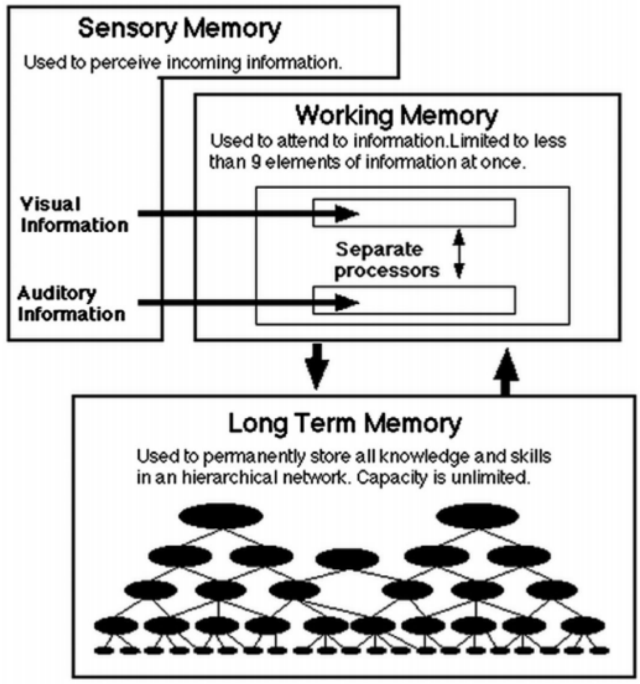


Figure 14: The CLT memory [Sweller 1988]

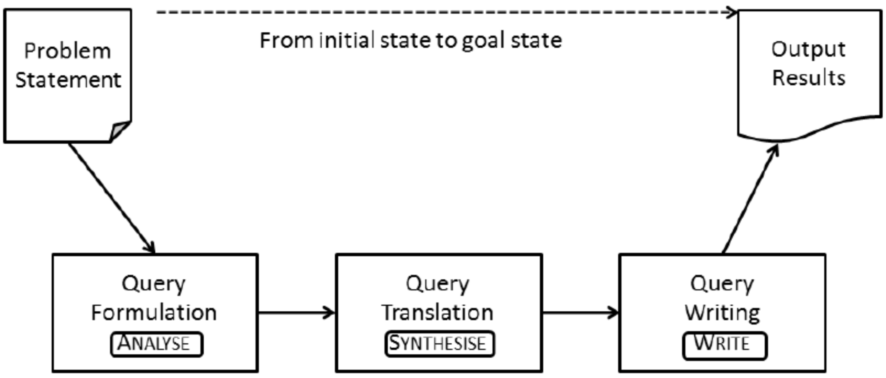


Figure 15: Problems to SQL generation [Al Shuaily and Renaud 2016b]

the program (Beacon). The top-down approach was first proposed by Soloway and Ehrlich [1984] to decompose programs into some levels of abstraction.

Khuziakhmetov and Porchesku [2016] presented the *bottom-up* program strategies based on developers reading code first and mentally grouping them. This strategy was referred to as the chunk model by Pennington [1987] and was used to group programs into a situation (or data-low) and program (or control-flow) model. Many learning theorists do not agree with either model (top-down or bottom-up), hence the *opportunistic* approach was introduced to allow the reading of only terms that are necessary in a program [Littman *et al.* 1986; Storey 2005].

The *knowledge-based* model, developed by Letovsky [1987], that combined both top-down and bottom-up approaches. The study described three components: mental model, which is a memory representation, knowledge that contains plans and goals, and an assimilation process. The integration of all approaches led to the introduction of the *integrated* model

[Von Mayrhauser and Vans 1995]. This model showed that switching between these models will improve the assimilation process of programmers.

3.6 COMPREHENSION AIDS: STATE-OF-THE-ART TOOLS

In recent decades, tools to aid the comprehension of SQL have been proposed. Most of these tools employ visualisation to explain how a query interacts with a database, providing interactive examples of the basic concepts of SQL. Others offer solutions to problems faced while learning SQL queries [Brusilovsky *et al.* 2008; Cembalo *et al.* 2011; Folland 2016]. In this section, we present some of the existing SQL tools from the earliest (that we know of) to the most recent that have been used in the comprehension of SQL.

3.6.1 *SQL Learning Tools*

3.6.1.1 *eSQL*

The `eSQL` system was proposed by Kearns *et al.* [1997] as an interactive learning system for students. It provides a step-by-step account of how a query result is determined, explaining each query step. For example, when a student writes a simple SQL query, `eSQL` provides a step-by-step formation of the resulting table. This ensures that even novices can easily grasp areas in SQL that are often found confusing. The authors stress that the dynamic stepwise mechanism provides a clear explanation of the underlying concepts of SQL, specifically allowing students to visualise the behaviour of query operators, and is far superior to the traditional pen-and-paper explanation approach.

Although, the `eSQL` system provides more information to a user with its friendly user interface, it does not provide comprehensive feedback based on the user's solution, due to the lack of semantic analysis in its engine.

3.6.1.2 *WinRDBI*

The `WinRDBI` system, also known as the Windows Relational DataBase Interpreter, is an educational tool that provides students with a friendly user interface to test their knowledge of SQL, relational algebra, and tuple relational calculus [Dietrich *et al.* 1997]. The system is ideal for most introductory courses on database management systems and provides a platform to get immediate feedback by seeing the answers to the query specified. One major limitation of `WinRDBI` is its inability to provide comprehensive feedback, much as with the `eSQL` system. Figure 16 presents the `WinRDBI` system.

3.6.1.3 *SQL-Tutor*

Mitrovic [1998] developed the `SQL-Tutor`, as an intelligent teaching system (ITS), for teaching SQL, implemented on SUN workstations. `SQL-Tutor` focuses on the SQL SELECT query and uses semantic analysis to provide feedback on query solutions. It is based on the constraint-based modeling (CBM), an approach focused on identifying and providing feedback on errors [Mitrovic 2003; Mitrovic and Ohlsson 2016]. An improved version of `SQL-Tutor` was introduced by Mitrovic [2003] known as `SQLT-Web`, it is a web version that addresses some of the shortcomings of the `SQL-Tutor`.

3.6.1.4 *AsseSQL*

`AsseSQL` was developed by Prior [2003] as an online assessment tool to test students' SQL formulation skills. It uses heuristics to evaluate whether a query entered is correct. The system is based on the SQL SELECT statement, and runs the submitted query on a test database. The query specified by a user is based on a question asked. `AsseSQL` compares the output of the query with the question. The goal of `AsseSQL` is to provide a deep learning experience for

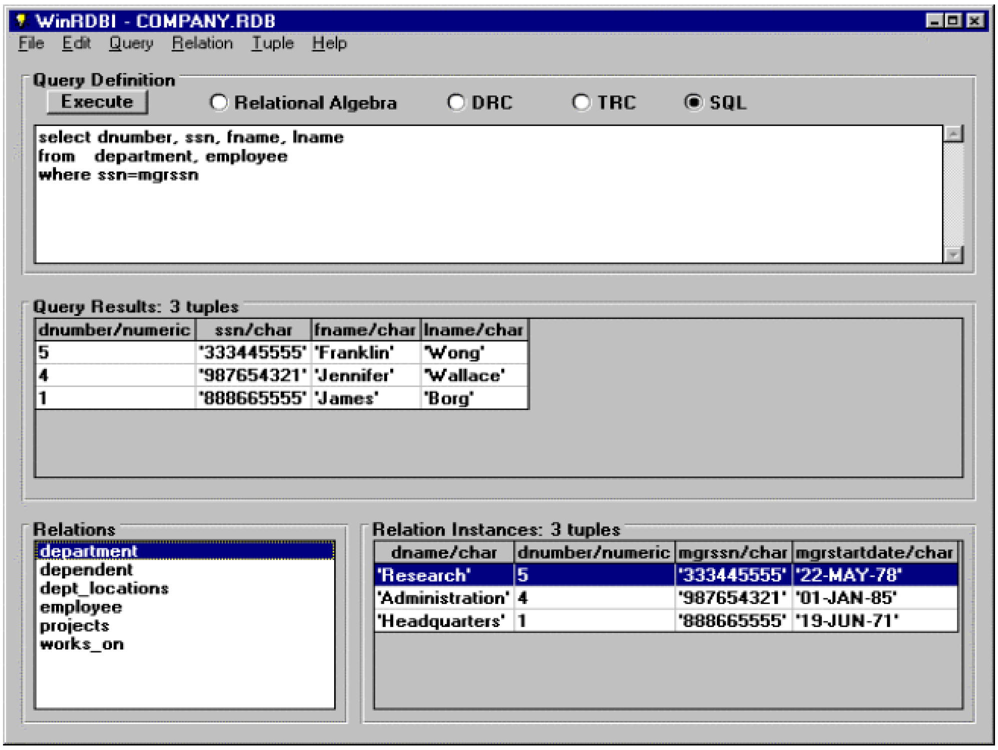


Figure 16: The user interface of the WinRDBI system [Dietrich *et al.* 1997]

students. Although AsseSQL was successful in providing feedback and grading options, it is vulnerable to SQL injection attacks – an attempt to make unauthorised changes to a database [Dekeyser *et al.* 2007]. An improved AsseSQL was introduced by Prior and Lister [2004] to evaluate users' perception of the system.

3.6.1.5 *SQLator*

The SQLator is a web-based interactive tool presented by Sadiq *et al.* [2004] at the University of Queensland for learning SQL. It uses heuristics as its engine to evaluate the correctness of formulated query. Much like the AsseSQL, it supports assessments and grading to queries submitted by students, but does not provide suggestions or hints to query formulation. SQLator has three main components, namely: *web application* – this is used for providing access to users, *engine* – implemented as a Microsoft COM[4] object for providing query formulation and evaluation, and *databases* – used for storing user data. Primarily, it supports only the SQL SELECT statement and judges if a proposed solution in SQL, corresponds to an English statement.

3.6.1.6 *SAVI*

SAVI, also known as SQL Advanced Visualisation, was created by Cembalo *et al.* [2011] as a system to aid the teaching and understanding of the semantics of SQL. It uses reversible animations to explain how query operators can transform data from a database. SAVI was implemented using the Google Web Toolkit framework written in Java, which allows Internet applications to be executed in any browser. The motive for this system was to help users overcome problems related to SQL and improve mental visualisation of query concepts. Although

4 Component Object Model

SAVI is efficient when using tables with a limited number of rows, it performs poorly with large tables.

3.6.1.7 *SiS*

SiS is an acronym for *SQL in Steps*. SiS is an online learning platform which allows students to learn and build SQL queries in a series of steps [Garner and Mariani 2015]. The goal of SiS is to improve the way in which users learn the SQL SELECT query by building a series of steps in the form of graphs. Hence, SiS is focused on the SQL SELECT statement because it is identified as an area of difficulty for students [Prior 2003; Sadiq *et al.* 2004; De Raadt *et al.* 2007].

3.6.1.8 *VisQlizer*

Folland [2016] proposed VisQlizer as a learning tool to help students create a mental model of the underlying concepts of SQL. VisQlizer uses animations and decomposition to aid comprehension. The researcher concluded that visualisation contributes to a better learning experience for students when coupled with traditional lectures and textbooks.

3.6.1.9 *QueryViz*

The query visualisation system (or QueryViz) was designed by Danaparamita and Gatterbauer [2011]. It reduces the workload needed to understand queries. QueryViz uses visual constructs to address issues students face with nested queries. As a learning tool, QueryViz allows novices to intuitively familiarise themselves with the logical patterns behind the SQL syntax using the visualisation.

3.6.1.10 *YASQLT*

Yet Another SQL Tutor (or YASQLT) is an automated assessment tool for SQL developed by Bider and Rogers [2016] that teaches the introductory aspect of SQL queries to novice students. YASQLT focuses on the SQL SELECT and CREATE VIEW statements. The goal of YASQLT is for checking the result of a query and addressing common errors made by novices while learning SQL. Lastly, the survey conducted with students using YASQLT showed that it was helpful in aiding their comprehension of SQL queries.

3.6.1.11 *OITS*

The Oracle Intelligent Tutoring System (OITS) is a tutoring system created by Aldahdooh and Naser [2017]. OITS automatically generates problems related to SQL queries to be solved by students. OITS consists of four basic components. These components are: Expert Module – responsible for identifying errors, Student Module – highlights the problem solving steps, Tutoring Module – keeps record of the student's progress, and UI module – integrates multimedia applications to aid learning. Empirical evaluation was conducted on OITS and the outcome concluded that the tool was friendly and easy to use.

3.6.1.12 *COSETTE*

COSETTE is an automated prover for SQL, developed by Chu *et al.* [2017], that can determine the semantic equivalence between two SQL queries. The main goal behind COSETTE is to determine if two SQL queries are semantically equivalent. COSETTE works with the SQL SELECT statements. In its metric, not all queries are supported. The study suggested that COSETTE can be used in a variety of real-world applications such as semantic caching, automatic grading and verifying the correctness of RDBMS rewrite rules.

3.6.2 *More Comprehension Aids*

In the previous section, we discussed and presented tools for aiding the comprehension of SQL. However, we cannot discuss all of them comprehensively. Other closely related tools developed to aid SQL are:

1. SQLify – A SQL teaching and assessment tool, developed by De Raadt *et al.* [2006], intended to offer a richer learning experience and provide comprehensive feedback to students.

2. LEARN-SQL – A learning environment for automatic rating of notions of SQL, presented by Abelló *et al.* [2008], that allows online assessment and learning in an interactive manner.

3. eledSQL – A web-based SQL learning tool proposed by Grillenberger and Brinda [2012], suitable for teaching SQL queries to novices.

4. SCYTHE – A web-based query-by-example system proposed by Wang *et al.* [2017a] that synthesises SQL queries from I/O examples.

5. SQL tester – An online practice aid developed by Kleerekoper and Schofield [2018] for assisting students to learn SQL queries, and providing immediate feedback.

6. SQL-to-text – A deep learning model using the graph-to-sequence approach proposed by Xu *et al.* [2018], which uses a graph encoder to generate SQL query to textual explanation.

7. GeoSQL Journey – A game-based tool designed by Sandoz *et al.* [2018] to stimulate student interest and simplify SQL learning.

8. RSQLG – A practice aid proposed by Julavanich *et al.* [2019], aimed at providing a hands-on environment to stimulate student's interest of learning SQL.

In this section, we have presented some SQL comprehension tools. The next section introduces NLP and its techniques.

3.7 NATURAL LANGUAGE PROCESSING AND TECHNIQUES

NLP has been applied in many fields such as CS, linguistics and cognitive science. In the next sections, we present the history and techniques used in NLP.

3.7.1 *Brief History of NLP*

NLP originated in the 1940s, just after World War II [Chowdhury 2003; Jackson and Moulinier 2007]. During this period, MT was the first language translation technique. At this time, people started to realise that a machine can be created to carry out this kind of task automatically. This phase was criticised due to primitive computing resources available [Hutchins 1986; Hutchins and Somers 1992]. This was the era of punch cards and batch processing that required no suitable higher-level language. The predominant language at this time was the assembly language, systems developed used dictionary-lookup for word re-ordering in a target language, which produced poor results [Bateman and Zock 2003]. NLP researchers realised that this task was more difficult than expected, hence there was a growing need to improve the theory of language [Lehnert and Ringle 2014]. Bates [1995] discussed that the first application of NLP was abandoned due to the complexity of getting computers to map one natural language (NL) into another. Thus, the study showed it was difficult to map one string to another.

From the 1950s to the late 1970s, speech and language processing was split into two paradigms called *symbolic* and *stochastic* [Jurafsky 2000]. The symbolic approach led to the work of Chomsky [1956], who introduced the idea of generative grammars and other formal methods. These methods were published in a book, *Syntactic Structures*, in 1957 [Chomsky and Lightfoot 2002]. In addition, many linguists and computer scientists started developing parsing algorithms: top-down, bottom-up and then dynamic programming [Roark and Johnson 1999; Kumar and Kanal 1988]. The development of parsing algorithms led to the earliest parsing system, which was the Zelig Harris's Transformation and Discourse Analyst Project (TDAP) [Heinroth and Minker 2012]. At this point, the linguistic field gained further insights on how to revive MT.

In 1956, the second aspect of the symbolic paradigm introduced the field of Artificial Intelligence (AI) [Shanmuganathan 2016; Mira 2008]. This field was based on the logic theorist perspective and the general problem solver model that created simple NLU systems. These NLU systems are based on pattern matching and keyword search, that were mostly used on QA systems. They use a knowledge-based approach that encodes knowledge and they are capable of producing answers for a provided question. The systems retrieve answers to questions from a database. Initially, LUNAR [Woods 1978] and SHRDLU [Winograd 1973] were the first NLIDBs that used this approach. The stochastic approach led to the development of the speech recognition engine, in particular, the Hidden Markov Model (HMM) began to show positive signs [Juang and Rabiner 1991; Sonnhammer *et al.* 1998]. This introduced works on speech recognition and synthesis.

From the 1980s to the late 1990s, NLP systems became accessible, and areas such as semantic-oriented processing tasks were built on NL systems [Sag *et al.* 2002; Ng and Zelle 1997]. By the end of the 1980s, statistical approaches were showing progress, which complemented some of the significant NLP problems already addressed by the symbolic approaches [Manning *et al.* 1999]. It became apparent that the NLP field was expanding. Most statistical methods such as probabilistic parsing were combined in machine learning to derive both syntactic rules and their probabilities [Hale 2001]. Even shallow processing methods such as finite state parsing and surface patterns were used in practical tasks [Grefenstette *et al.* 2000; Roche and Schabes 1997]. These methods were extended to dialogue structures in conversational systems especially in gaming technologies. Similarly, algorithms for discourse processing, reference resolution, parsing and part-of-speech (PoS) tagging began to incorporate probabilistic measures into their methods [Ng and Low 2004; Nieuwland *et al.* 2007]. There was an increase in the computer memory and speed of processing systems, which allowed for commercial implementation of subareas of NLP such as speech recognition. Grammar and spelling algorithms began to apply augmented alternative communication [Garcia-Molina and Salem 1992; Myers 1998]. All these contributed to the rise of the Web with needs for language-based information retrieval and extraction methods [Srinivasan and Brown 2002]. These methods extended to the growth of linguistic resources such as WordNet [Miller 1995], British National Corpus [Leech 1992] and test tools such as the Penn Treebank Tagset [Marcus *et al.* 1994] was developed.

The early 2000s saw the development of the first neural language model proposed by Yoshua Bengio and his team [Bengio *et al.* 2003; Bengio and Senécal 2008]. The neural network describes an artificial neural network, an aspect of the deep learning model that moves data in one direction through hidden nodes and extends this to an output node. More neural approaches have been developed, and extended to problems related to NLP [Collobert and Weston 2008; Collobert *et al.* 2011]. Notably, many computational problems such as word sense disambiguation, and part-of-speech identification have become standards, which are used throughout NLP [Warner and Hirschberg 2012; Li *et al.* 2010]. Currently, popular NLP powered tools such as Apple's Siri, Amazon's Alexa and Google's Home have been developed, which are used on most portable devices [Krusche *et al.* 2018; Kugler 2019].

3.7.2 *Classification of NLP*

NLP is a combination of AI and Linguistic methods, and aims to make computers understand human languages. According to Manning [2014], NLP is classified into NLU and NLG. Jusoh and Alfawareh [2012] described NLU as a representation that deals with modeling human reading comprehension tasks which parses and translates inputs according to NL principles. Typically, this uses NLP algorithms to reduce human speech into structured forms. NLU consists of components such as phonology, morphology, pragmatics, syntax and semantics. Reiter and Dale [2000] described NLG as a technique where texts are generated from human languages using computer-accessible data. This process understands texts in natural form such as English from non-linguistic representation of information [Manning 2014]. These terms are illustrated in Figure 17. We describe each of these terms.

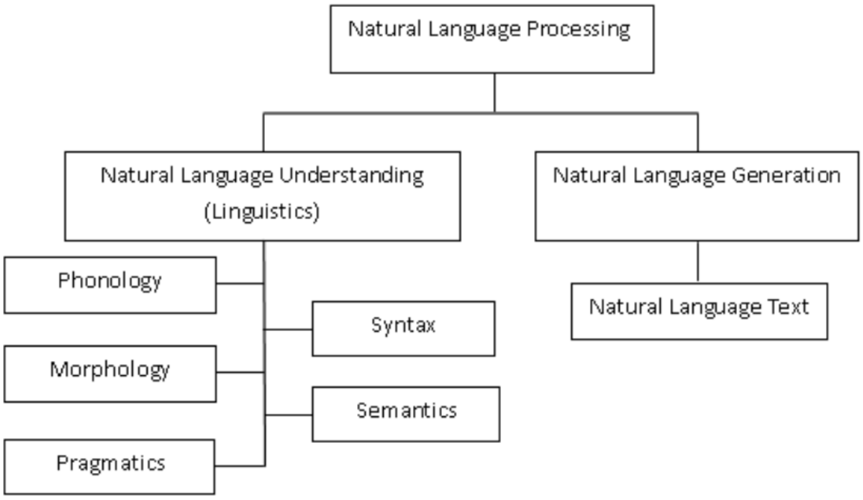


Figure 17: Classification of NLP [Khurana *et al.* 2017]

PHONOLOGY The field of phonology deals with the study of sounds and how they are used in languages [Gussenhoven and Jacobs 2017]. Phonology is applied to virtually all languages, and is predominantly used in the linguistic domain. In the computational discipline, phonology refers to the application of computational techniques to the processing of phonological information [Bird 2002]. Trask [2004] described that phonology tells us what sounds are contained within a language and shows what happens if they are combined into words. The study further described that phonology and phonetics are two sub-disciplines in linguistics and highlighted that speech organs and muscles are involved in the different aspects of a language. The human phonological features differ with varying frequencies according to their supralaryngeal vocal tract. This is depicted in Figure 18.

MORPHOLOGY The morphology of a language describes how words are put together to form a grammar [Ritchie *et al.* 1992]. Words are an essential part of linguistics, and constitute an integral part of mental grammar [Twain 2013]. A native speaker of a language is expected to know thousands of words. Nordlinger and Sadler [2019] noted that without words, it will be difficult to convey thoughts through a language or understand others' thoughts. Hence, a native speaker of English knows how to segment sounds of words in his or her lexicon. In NLP, morphology is important for lemmatisation and parsing. They are useful in many applications [Twain 2013; Nguyen *et al.* 2016]. For example, morphology has been applied in lexical databases such as WordNet [Miller 1995] and Arabic Ontology [El-Affendi 2018], and statistical parsing tools such as SPRML [Tsarfaty *et al.* 2010] and FUNGuild [Nguyen *et al.* 2016].

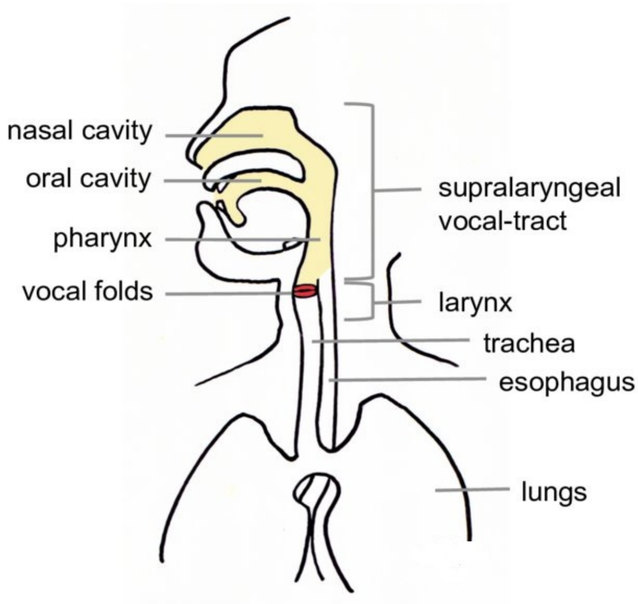


Figure 18: The human vocal apparatus [Pisanski 2014]

PRAGMATICS Pragmatics is a branch of linguistics that deals with how contexts can be converted into meanings through a language [Thomas 2014]. Pragmatics has its roots in sociology, anthropology and philosophy [Morris 1970]. This study was influenced by the work of Peirce [1902], who described three systems of signs (semiotics) namely, syntax, semantics and pragmatics. The study defined *syntax* as the formal relation of signs, *semantics* as the relation of signs to what they denote, and identified that *pragmatics* is used to describe signs with relations to users and interpreters. Many NLP researchers discussed that, to produce effective NLP systems, it is important to understand the pragmatics of a natural language [Cruse 2011; Ward 2016]. Cherpas [1992] stressed that semantics blends well with pragmatic theory, as it enables the recognition of human conversations, and how they can manipulate each other. In addition, the history of their interaction can shape future utterances.

SYNTAX Covington *et al.* [1994] described syntax as the set of rules that governs the formation of words into phrases or sequences of well-formed words (sentences). The goal of syntax is to relate morphological components to semantic constituents [Chakrabarti 2004; Dale *et al.* 2000]. In other words, syntax involves word tokens and structure. In syntactical forms, words combine in a way which mirrors the expected meaning. For example, *Peter loves Mary might indicate something different from Mary loves John*. There is usually ambiguity involved at the syntatic phase [Manning *et al.* 2010]. To represent these forms, syntax trees are usually used to denote the syntactical analysis phase [Van den Brand *et al.* 2005; Hewitt and Manning 2019]. In NLP, syntax is regarded as a lower level stage and involves the following activities [Nadkarni *et al.* 2011; Room 2019; Gill 2019]:

1. Lemmatisation: This process involves reducing words to their base form, known as *lemma*.
2. Morphological Segmentation: This process breaks words into morphemes. For example, the English word, "horses" contains two segments (horse) + (s). This is a word stem and its suffix.
3. Word Segmentation: This is the process of dividing strings in a language into component words.
4. Normalisation: This process involves the categorisation of words or tokens into a standard format.

5. Stemming: This is the process of reducing words within the same stem to their root form.
6. PoS Tagging: This is the process of parts of speech identification for each word. For example, the PoS for "Johannesburg" is a noun.
7. Parsing: This is the grammatical analysis of a given sentence into a syntax tree.

SEMANTICS Berant and Liang [2014] define semantics as a concept that conforms to meaning. The process of checking if words form sensible sets of instructions is known as *semantic analysis* [Nasukawa and Yi 2003]. At the semantic analysis phase, this process relates the syntactic structures from granular levels of phrases, clauses to the language independent meaning [Cambria and White 2014; Sun *et al.* 2017b]. This has been applied in many domains such as: Medicine [Pons *et al.* 2016], Affective Computing [Cambria 2016] and Game theory [Ryan *et al.* 2015], etc. In NLP, if a language is to be understood by a computer, it must go through syntactic and semantic analysis phases [Seuren 2017]. The upper level activities of NLP as described by Nadkarni *et al.* [2011]; Khurana *et al.* [2017] are:

1. Named Entity Recognition: This task involves determining the pre-defined categories of named entities such as names of organisation, persons, locations, etc.
2. Word Sense Disambiguation: This task determines the meaning of an ambiguous word within a context.

3.7.3 *NLP Algorithms*

Since the inception of NLP, different algorithms have been described to work with tools in this field. Duh [2018] describe NLP algorithms as useful in multiple language variations. In this section, we review the NLP algorithms that have been proposed over the years.

3.7.3.1 *Naïve Bayes*

Naïve Bayes (NB) algorithm has emerged as one of most efficient and effective text classification techniques used for data mining and machine learning tasks [Gao *et al.* 2018; Li *et al.* 2018; Xu 2018]. This classification technique was developed by Thomas Bayes in 1763, hence its name [Wang *et al.* 2016]. It was regarded as *naïve* because it assumes features that are used by a machine learning model which are independent of each other. Although currently dubbed as 'the punching bag' of newer classifiers by machine learning theorists, it remains widely used for classification of texts, which is fast and easier to implement [Xu 2018; Mohammed *et al.* 2017]. Most spam filters include NB in their commercial and open-source projects. In addition, it has been applied in many real-world classification problems in medicinal diagnosis, sentiment analysis, weather predictions, etc [Xu 2018; Wood *et al.* 2019; Kwon *et al.* 2019].

Li *et al.* [2018] described NB as a probabilistic classifier that uses the Bayes theorem which can substitute logistic regression models, and can also be used to formulate dependency or independent variables. Furthermore, Bayes theorem is also referred to as a posterior probability, used for an event [Zhang *et al.* 2019a].

In a comparative study, Keogh [2006] showed that NB was much faster to train, and insensitive to irrelevant features in data. A major flaw identified in the study showed that NB assumes independence on more linear features than on non-linear features. Furthermore, Xu [2018] discussed that the NB's classical counterparts such as the Hidden Markov Models (HMMs), Support Vector Machine (SVM) and principal component analysis (PCA) were much better for text classification compared to the NB.

3.7.3.2 *Support Vector Machine*

The SVM is a widely text supervised model proposed by Boser *et al.* [2003] that has been applied to numerous real-world problems. SVM belongs to a family of linear classifiers used for classification and regression tasks [Ben-Hur and Weston 2010]. As a technique, it is regarded as one of the kernel methods that maximise predictive accuracy while avoiding over-fit to data [Ben-Hur and Weston 2010]. These method provides two advantages: they have the ability to generate non-linear boundaries and they allow a classifier to use data with no obvious fixed space representation. This has been applied in protein synthesis, Deoxyribonucleic Acid (DNA) sequences and bio-informatics problems [Shawe-Taylor *et al.* 2004].

One simple rule about SVMs is that they create a line (or a hyperplane) that separates data into classes, which can be applied to many classification problems [Evgeniou and Pontil 2001; Tang 2013]. Klein [2006b] explained the hyperplane using an illustration. In Figure 19, the first hyperplane, H_1, is indicated as a black circle which incorrectly classifies data points. The second and third hyperplanes, H_2 and H_3, correctly classify data points.

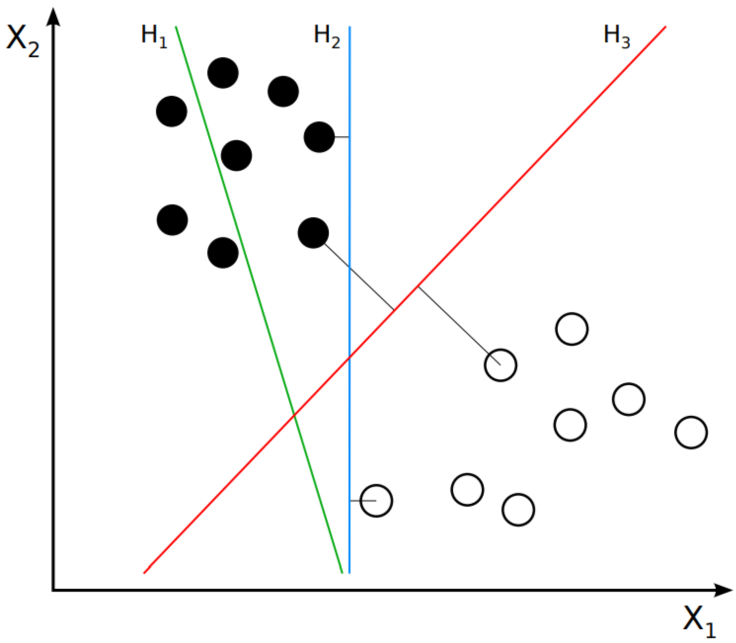


Figure 19: Support Vector Machine: Separating Hyperplanes [Klein 2006b]

Auria and Moro [2008] identified the strength and weaknesses of SVM. Notably, SVMs are good with structured and unstructured data e.g. text, trees and images. One major drawback of SVM is that it takes longer to train a large dataset. Goldberg and Elhadad [2008] noted that despite the challenges faced with SVMs, they contribute immensely to solving problems in the NLP domain.

3.7.3.3 *Hidden Markov Model*

One of the most popular methods used in machine learning for sequence modelling of speech and protein is the HMM [Tobon-Mejia *et al.* 2011]. Since introduced by the Russian mathematician, Andrey Andreyevich Markov, as the *theory of stochastic Markov processes*, it has contributed immensely to solving many real-world problems [Van Kasteren *et al.* 2010; Narasimhan *et al.* 2016; Fu *et al.* 2016]. Almost all modern speech recognition systems are based on HMM. Although, its framework has not changed drastically, it has evolved even further with more sophisticated features [Gales *et al.* 2008]. Bahdanau *et al.* [2016] described the HMM as a double stochastic model that is based on intrinsic variability of spectral features and a statistical modeling framework that ensures consistency in detecting spoken languages. These stochastic processes are characterised by states and transitions probabilities. The first process, states, are not visible, hence it is considered to be *hidden*. The second process, transitions, attempts to produce state-dependent probability distributions [Kouemou and Dymarski 2011].

Keselj [2009] described the HMM as based on the augmentation of the Markov chain, which shows the probabilities of sequences of random variables. The study further explained that for a NLP scenario, HMM shows observed events and hidden event (PoS tags) that are considered in a probabilistic model. Furthermore, Awad and Khanna [2015] explained that for given observations, HMM is used to describe the solution to a problem in a state-sequence determination manner and through model training means.

3.7.3.4 *Deep Learning*

The era of deep learning methods began in the 20th century after the performance of traditional learning became less satisfactory to process human information forms in speech and vision [Ohlsson 2011; Sun *et al.* 2017a]. The deep learning methods were originated by Geoffrey Hinton in 2006, when he proposed the Deep Belief Network (DBN), a deep learning structured architecture [LeCun *et al.* 2015]. Since the DBN, there have been rapid developments of other deep learning techniques with significant impacts on information processing in the medical, engineering, and even the educational fields [Ching *et al.* 2018; Lim *et al.* 2016]. Predominantly, the deep learning methods have contributed immensely to the NLP [Socher *et al.* 2012; Deng *et al.* 2014; Bian *et al.* 2014], speech recognition [Hinton *et al.* 2012; Amodei *et al.* 2016] and computer vision [Kendall and Gal 2017; Voulodimos *et al.* 2018]. After the DBN network, Artificial Neural Networks (ANNs) became popular and were an active area of research, utilising neurons to produce real-valued activation. These neurons receive inputs from external sources, and assign weights and biases during training. Then, it produces an output. This is illustrated in Figure 20. The diagram shows the input layer in the leftmost layer, the hidden layer indicates that it is neither an input nor an output medium. The output is the rightmost layer or output neurons.

Comparable to the ANNs are the Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). They are comprised of neurons that receive an input and perform a scaler operation to produce an output [Goodfellow *et al.* 2016]. CNNs are mostly used for pattern recognition within images and have recorded major successes [Khosravi *et al.* 2018; Sturmfels *et al.* 2018]. RNNs are popularly used for sequential tasks such as speech and language (texts) [Di Persio and Honchar 2016; Goodfellow *et al.* 2016]. RNNs are powerful networks and training them requires backpropagation which exhausts a great deal of computational power. Together, these neural networks have performed well on various NLP tasks such as named-entity recognition, MT, phrase detection and language modeling [Manning 2015; Zheng *et al.* 2013]. One major contributing factor of these neural networks is their ability to perform tasks without time-intensive engineering processes. This has led to an important concept in NLP, known as Word Embedding [Weston *et al.* 2012]. In addition, a survey study conducted by Këpuska and Bohouta [2017] showed that due to the deep learning method integrated in the Google Speech recognition engine, it improved its word error rate (WER) from 23% in 2013 to 8% in 2015. The study showed that Google's speech engine was significantly better compared to the Microsoft Speech API and Sphinx-4 engine.

3.7.4 *Popular NLP Libraries*

NLP libraries are mostly used by researchers to extract information from texts [OpenNLP 2011; Manning *et al.* 2014; Zhang *et al.* 2019b]. These libraries handle a wide range of NLP

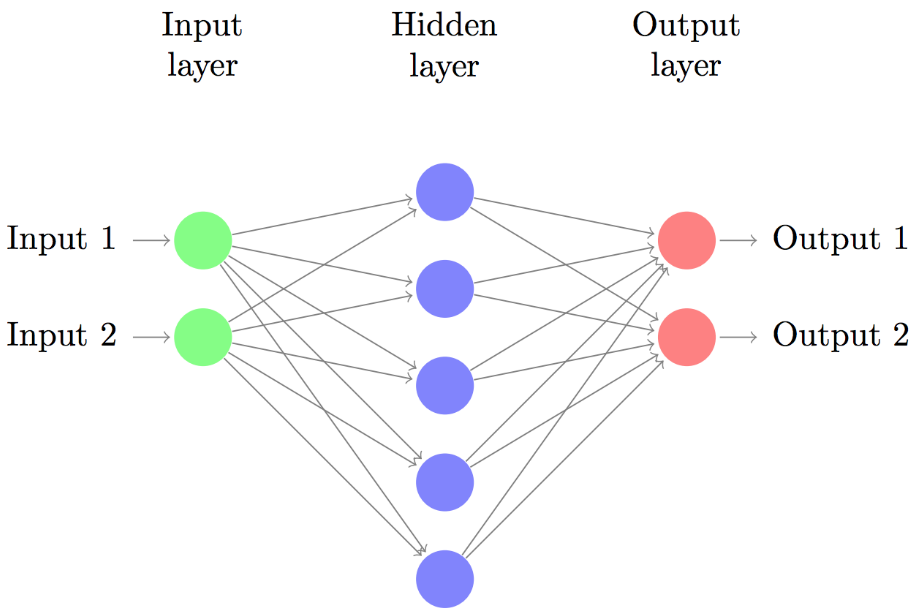


Figure 20: A neural network [Liu *et al.* 2017]

tasks such as topic modelling, text classification, sentiment analysis, POS tagging and many more. In this section, the libraries used for NLP tasks are presented.

3.7.4.1 *Apache OpenNLP*

The Apache OpenNLP[5] was written in Java, as a free, open-source machine learning tool used for core NLP tasks, such as named entity recognition, parsing, sentence segmentation, and POS tagging [Baldridge 2005; OpenNLP 2011]. Developers use the OpenNLP interfaces, provided by means of an API[6] to implement these NLP tasks. The OpenNLP library uses a maximum entropy to build advanced text processing services [Dandapat 2007; Tratz *et al.* 2007; Bilgin 2019]. The maximum entropy framework is based on the principle of making assumptions based on constraints imposed on training data, relationships between data features and expected outcome. In addition, the maximum entropy is used to recognise different entities such as locations, organisations, dates and persons. The OpenNLP has been applied to solve many problems in different domains. We describe them in no particular order.

Rodrigues *et al.* [2018] used the OpenNLP tool for NLP tasks specified in the Portuguese language. In this study, sentences were split into tokens, which are a combination of words. These words are further analysed to broaden the result. The study reported that while using OpenNLP, many tools under-performed in some language constructs within the Portuguese context. During an analysis of the forum posts on the dark web, Park *et al.* [2016] applied the OpenNLP for sentiment scores generation. The study used the OpenNLP tool tokeniser feature to create an array, which is then used to find all parts-of-speech in each post, and generate frequencies of each noun in the post. In the biomedical field, Zhang *et al.* [2019b] developed a tool called RNA Interactome Scoper (RIscoper) that uses the OpenNLP for sentence seg-

5 https://opennlp.apache.org/
6 Application Programming Interface

mentation which extracts contents from articles. According to the study, the developed tool will save time required for drafting of literature reviews and data organisation in databases. In addition, the study emphasised that the tool would be useful for bioinformaticians and experimental biologists.

3.7.4.2 *Natural Language Toolkit*

The Natural Language Toolkit[7] (NLTK) provides a set of tools, released under an open-source licence for performing different NLP tasks [Loper and Bird 2002; Bird *et al.* 2008; 2009]. NLTK was developed in Python and contains libraries that supports statistical and symbolic NLP. Python was selected as the implementation language because its syntax and semantics have good string handling functionality and provide a shallow learning curve [Bird *et al.* 2008]. This toolkit has comprehensive documentation, including tutorial guides that contain the processing tasks it supports. Predominantly, NLTK is well suited to users learning or conducting research in NLP including related areas such as machine learning, cognitive science and information retrieval [Klein 2006a].

Bird *et al.* [2009] described that before NLTK was developed, the following goals were kept in mind. The first goal is *simplicity*, an intuitive framework that could give users a practical knowledge of NLP was created. Second, there was a need to ensure *consistency* with interfaces that provide a uniform framework. Third, there is a need to provide a structure that allows new modules to be *extensible*. Finally, the study showed that some components can be used *independently* of other components. Lobur *et al.* [2011] highlighted the uses of NLTK, which includes chunk parsing, assignments, as well as advanced tasks such as word sense disambiguation and morphological analysis.

3.7.4.3 *Stanford CoreNLP*

The Stanford CoreNLP[8] toolkit is an extensive annotation pipeline framework, developed in Java, that provides features to most NLP tasks from tokenisation, named-entity recognition through to coreference and basic dependencies [CoreNLP 2016; Hirschberg and Manning 2015; Angeli *et al.* 2014]. Manning *et al.* [2014] described the toolkit as a combination of multiple components, each with their APIs tied together to function as a custom glue unit (or code). Before using the engine, raw text is inserted into some annotated object, which then undergoes various NLP processing tasks and the resulting feedback is provided by means of an annotated or plain text format. We describe some of the recent works that have extended the Stanford CoreNLP into other languages.

For recognising lexicons within the Chinese language, Peng *et al.* [2015] developed a tool called *CONCRETE*, an NLP pipeline built on a number of open-source tools. This pipeline extended the CoreNLP framework to recognise Chinese language within a broader context. The pipeline supported word segmentation, named entity recognition, parsing and PoS tagging. Bondielli *et al.* [2018] developed an extension of the Stanford CoreNLP toolkit based on the Universal Dependency (UD) framework into a tool, called the `CoreNLP-it`. This tool was developed for the recognition of the Italian language as a set of customisable classes. This study reported that the tool was UD compliant, catered for multi-word token representation and provided an extensible framework to support other languages. Andreeva *et al.* [2018] extended the CoreNLP toolkit to recognise texts in the Russian language. This tool was able to determine parts of speech in the Russian language and this proved effective in this context.

7 http://www.nltk.org/
8 https://stanfordnlp.github.io/CoreNLP/

3.7.4.4 *spaCy*

The spaCy[9] NLP engine was developed as an open-source Python toolkit, designed to work on large-scale commercial information extraction tasks [Al Omran and Treude 2017; Bocklisch *et al.* 2017; Srinivasa-Desikan 2018]. The current version, spaCy v2.x, was developed in Python/Cython in 2017, which possesses an accuracy of 92.6%, making it the fastest syntactic parser in the world. Till now, spaCy is the fastest NLP toolkit in the world with regards to language processing tasks, especially when compared to NLP libraries that have been developed [Al Omran and Treude 2017]. spaCy supports almost all NLP tasks from dependency parsing, tokenisation to POS tagging, and it works well with most deep learning libraries such as scikit-learn, TensorFlow, PyTorch [Jangid *et al.* 2018; Goyal *et al.* 2018]. We describe recent works that have applied the spaCy toolkit to their tools for various NLP tasks.

Bocklisch *et al.* [2017] built a tool called `Rasa`, an open-source Python framework, used to build conversational systems using the spaCy toolkit for NLP tasks. For NLU tasks, the spaCy toolkit was used to perform tokenisation and POS tagging. Kejriwal *et al.* [2017] extended the use of the spaCy framework into an open-source tool called `FlagIt`, a system for mining problems in the sex trafficking domain. The system has been integrated into a domain-specific search platform used by over 200 law enforcement agencies to minimise the problem of human trafficking. The study noted that spaCy was effective in the inherent complex NLP tasks used by `FlagIt`. In a study on grammatical error corrections, Náplava and Straka [2019] developed a tool called the `CUNI` system that applies restricted, unrestricted and low-resource tracks trained using the Wikipedia source to resolve errors in a sentence. `CUNI` applied spaCy to correctly tokenise sentences. The study showed that spaCy was effective in this task.

3.7.5 *Applications of NLP*

NLP is one of the most important technologies of the current information age [Klein *et al.* 2017; Nakazawa *et al.* 2006; Jurafsky and Manning 2012]. As humans communicate their thoughts in a language, this has paved the way for numerous applications of NLP: language translation, web search, advertisements, spam detection, text categorisation, and QA. In this section, a few NLP application areas are discussed.

3.7.5.1 *Machine Translation*

Sokolov *et al.* [2016] described MT as an important application area of NLP that deals with the automatic translation of speech or text from one human language into another. Hirschberg and Manning [2015] emphasised that MT is the most substantial way in which computers could facilitate human to human communication. MT is rated as the most difficult field in NLP, because it relates to human lives, and as such, it had become a million dollar affair [Alsohybe *et al.* 2017].

Over the past decades, MT has been an active area of research for linguists, computer scientists and engineers [Klein *et al.* 2017; Nakazawa *et al.* 2006; Costa-Jussa and Fonollosa 2016]. It is interesting to note that the first of numerous applications of computers was the MT. This was studied intensively in the late 1950s [Hirschberg and Manning 2015]. This era saw the development of the Statistical Machine Translation (SMT) and Rule-based Machine Translation (RMT). In the early 1990s, the MT field was transformed at the bilingual Canadian Parliament proceedings when there was parallel text translation of English and French sentences [Jurafsky and Manning 2012]. Since then, there has been improvement in language translation in the MT field, and currently, the field is in a state of flux with hybrid

9 https://spacy.io/

solutions, falling short of precision and accuracy of human translators [Dale 2019].

Koehn *et al.* [2007] presented an open-source toolkit called `Moses` for SMT that consists of components for data preprocessing, language model training and result translation. This toolkit integrates well with NLP/speech processing tools with varying confidence in a consistent and flexible framework. The study concluded that `Moses` will be of immense value to the MT community. The advancement of the deep neural network has paved the way for Neural Machine Translation (NMT), which is a recently proposed approach to the MT field [Koehn and Knowles 2017; Artetxe *et al.* 2017; Zoph *et al.* 2016]. Unlike the SMT approach, NMT builds on a single neural network that maximises translation performance. In addition, the SMT field is problematic because most translation systems are specifically trained within a particular domain [Farajian *et al.* 2017]. Thus, it might perform poorly in a different domain. With the introduction of NMT, state-of-the-art systems perform better in English-French translation tasks [Bahdanau *et al.* 2014].

3.7.5.2 *Question Answering*

QA is an active area of research and a specialised type of information retrieval (IR) or information extraction (IE) aimed at returning answers to queries presented in the form of a natural language [Ong *et al.* 2009; Belinkov *et al.* 2015; Cantador *et al.* 2011]. Athenikos and Han [2010] discussed that the next generation of search engines will utilise the capabilities of QA. The history of QA systems dates back to the late 1960s and early 1970s when a major surge of research activities was seen within the IR/IE community [Athenikos and Han 2010]. In 1999, this surge led to the establishment of the QA Track in the famous Text REtrieval Conference evaluations [Voorhees and Harman 1999]. Since then, a number of techniques has been developed for answer generation for three questions types such as list, factoid and definitions [Athenikos and Han 2010]. To support QA systems, numerous large datasets have been developed over the years. These datasets consist of questions posed by crowdworkers[10] that are used to train QA systems, where the answer is a segment of text within the content. Examples are `SQuAD` [Rajpurkar *et al.* 2016], `HotpotQA` [Yang *et al.* 2018], `DAWQAS` [Ismail and Homsi 2018], `30M Factoid questions` [Serban *et al.* 2016b] and many others.

Generally, QA systems consist of three processing methods, namely: question, documents and answering phases [Hirschman and Gaizauskas 2001]. They are built with semantic knowledge throughout the QA process, in order to derive correct answers to questions. The semantic information is obtained from questions and ontological resources may be used to improve the performance of the QA system. According to a review study by Athenikos and Han [2010], QAs were classified into semantic-based QA systems, inference-based, and logic-based. The study concluded that QA systems will continue to grow and help users better utilise the evolving nature of information.

3.7.5.3 *Spam Detection*

There is a no single, accepted definition of spam, although this problem has gained prominence since the 1990s [Van Wanrooij and Pras 2010; Chandra and Suaib 2014; Hayati *et al.* 2010]. Iedemska *et al.* [2014] described spamming as an activity perpetuated by cybercriminals, which is used to generate income to the tune of millions of dollars. Conversely, spam may contain unsolicited advertising contents [Broadhurst and Trivedi 2018]. Spamming activities are conducted using platforms such as websites reviews [Lin *et al.* 2014; Ghai *et al.* 2019; Saini *et al.* 2019], social media [Barber *et al.* 2018; Sharaff *et al.* 2016], opinion mining [Rayana and Akoglu 2015; Chen and Chen 2015] and email [Idris *et al.* 2015; Seyyedi and

10 a method that involves volunteers to accomplish a specific task

Minaei-Bidgoli 2018] to swindle users. Email spamming appears the most popular amongst these activities [Christina *et al.* 2010]. This reduces human productivity, wastes bandwidth and storage, and has exceeded legitimate emails which are sent over the Internet [Shue *et al.* 2009].

Over the past few years, numerous studies have been proposed to detect spam. We present the recent studies in no particular order. Using NLP algorithms, Ezpeleta *et al.* [2016] proposed the use of the Bayesian filtering classifier to detect unsolicited emails, which are major threats affecting millions of users per day. The research achieved an accuracy of 99.21% exceeding those proposed by machine learning algorithms. Similarly, Maguluri *et al.* [2019] extended the use of Bayesian classification to email spam detection. The study categorised email messages as either *spam* or *non-spam* and noted that spam could be an enormous problem for private and public organisations. Classification techniques such as SVM and Deep Neural Networks have also proved effective to efficiently identify spam in emails [Torabi *et al.* 2015; Schölkopf *et al.* 2002; Agarwal and Kumar 2016].

Kumaresan and Palanisamy [2017] applied the SVM technique to detect spam emails, showing a better accuracy over other techniques due to the small data size. The study reported an accuracy of 97.235% when compared with an existing approach. Roy *et al.* [2019] extended the deep networks using CNN and the Long Short Term Memory (LSTM) to detect spam in a social network such as Twitter. The study incorporated semantic databases such as WordNet and ConceptNet to improve semantic information representation. Furthermore, this improved the accuracy and F1-score of the result.

3.7.5.4 *Dialogue Systems*

Recent advances in NLP have led to multiple applications of dialogue systems, which have significantly eased tasks in medicine, online shopping, technical support, etc [Deng *et al.* 2013; Serban *et al.* 2016a; Bowden *et al.* 2019]. These dialogue systems provide either *speech* or *type-written* features, or both [McTear 2002; Glas *et al.* 2012]. Speech dialogue systems are interactive platforms used by humans to communicate with a computer with the intention of achieving a specific objective [Serban *et al.* 2016a]. For example, a user may request all hotels that are based in a particular location from a chatbot[11]. The chatbot takes the request and provides immediate feedback to the user. With the growing rate of AI, numerous companies are beginning to design powerful goal-oriented spoken dialogue systems. Examples of such spoken dialogue systems are Google's Assistant, Apple's Siri, Amazon's Alexa and Microsoft's Cortana [López *et al.* 2017; Hoy 2018]. Chen *et al.* [2017b] described the components required of any speech dialogue system. These components are automatic speech recognition for recognition of human speech into text, NLU framed into speech identification, dialogue management using backend providers and NLG for generating texts based on some linguistic methods.

Nowadays, a number of NLP libraries have been developed, which are used by dialogue systems for communication. Examples of these libraries are OpenDial [Lison and Kennington 2016], AllenNLP [Gardner *et al.* 2018], ParlAI [Miller *et al.* 2017]. General frameworks such as Stanford CoreNLP and spaCy have been used for the development of dialogue systems [Burtsev *et al.* 2018; Li *et al.* 2016].

The origin of dialogue systems can be found in tools such as `ELIZA` [Weizenbaum and others 1966] designed to allow humans to interact with computers in free-forms, specified in natural language. Similarly, other tools such as `Parry` [Colby 1975] and `Alice` [Wallace 2009]

11 an AI-powered system designed to simulate conversation with human users

were developed to allow users interact in a conversational manner. Although these tools were able to perform tasks seamlessly, they were not intelligent enough and lacked the ability to keeping conversations evenly [Shum *et al.* 2018]. Hence, they only work well in constrained environments. Since then, many researchers have developed smart solutions on dialogue systems that can handle complex tasks. Huang *et al.* [2015] developed a spoken dialogue system called `Guardian` that significantly can improve interaction in a cost-effective manner. The tool combines expert and non-expert processes that uses a Web API to scale up interactions, and can be embedded in other dialogue systems through its API features. Similarly, a tool called LS-SDS was developed by Papangelis *et al.* [2017] as an advanced, complex, interactive interface that leverages on Linked Data to improve the linking of entities to user's input and data sources. Goel *et al.* [2019] proposed a hybrid approach that uses a trainable neural network coupled with a NLU and dialogue tracking to achieve a high accuracy. This hybrid approach helps to facilitate interaction between a user and a computer system.

3.7.5.5 *Text Categorisation*

In the data-intensive application fields such as banking, universities and funding agencies, unstructured data remains a serious problem [Tang *et al.* 2016; Selvi *et al.* 2017]. One way of tackling this problem is to present the data in a format that can be used in these fields [Al-Radaideh and Al-Abrat 2019]. This process is regarded as text categorisation (or text classification). Dumais *et al.* [1998] described text categorisation as an application of NLP, that deals with the assignment of natural language texts to one or more categories based on their linguistic contents. Sebastiani [2005] explained that text categorisation involves the task of automatically sorting document sets into a set of categories. This area of NLP has received prominence in the last 10 years, and many researchers and software developers are deploying applications using this approach. There are well-known text classification methods such as language detection, sentiment analysis and topic labeling [Stein *et al.* 2019; Huang *et al.* 2014a; Chaturvedi *et al.* 2018]. These methods have been used to solve numerous real-world problems.

For a classification of web page content, Qi and Davison [2009] described state-of-the-art practices as essential to many tasks such as information retrieval, web directory maintenance and data crawling. The study showed that classification can improve web search quality. The four classification techniques as described in Figure 21, show how they can used to resolve search engine spams and improve search results in websites. For clinical data classification problem, Bui and Zeng-Treitler [2014] used the regular expression approach alongside the SVM classifier for text classification of clinical datasets. The study reported that using these two classifiers improved clinical text classification performance and showed that this hybrid approach was significantly better than using the SVM classifier alone. Hughes *et al.* [2017] presented an approach for the classification of clinical text at the sentence level. The method used the CNN approach to train the health information dataset and indicated that the approach superseded previous approaches that have been developed by about 15%. In a similar approach, Chen *et al.* [2017b] used a CNN model to classify radiology reports with accuracy reaching 99%. The study reported that this approach may be used for large-scale applications by annotating texts in medical imaging reports.

3.7.6 *Classification of Natural Language Interfaces to Databases*

Mony *et al.* [2014] described NLIDBs as systems which provide easy access to databases using a natural language, without requiring a user to write in a query languages such as SQL, Prolog and Lisp. These systems are mostly used by non-technical end-users in fields such as banking, medical, engineering, mining, etc [Yuan *et al.* 2019; Gantayat *et al.* 2019; Kapetanios 2008]. To use NLIDBs, Wudaru *et al.* [2019] emphasised that users are required to communicate with these systems in a natural language such as English. Other languages specified in Arabic [El-Sayed 2015] and French [Etzioni and Kautz 2002] have been proposed in the past. The develop-

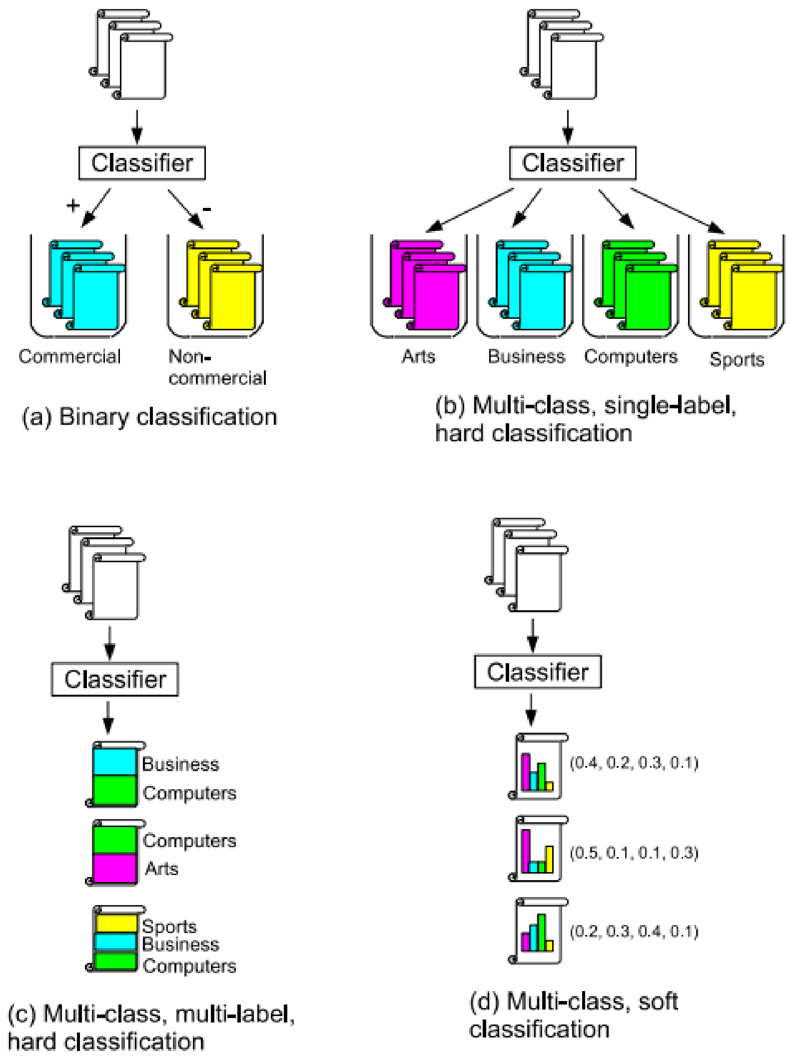


Figure 21: Text classification methods for web documents [Qi and Davison 2009]

ment of NLIDB systems started in the 1960s [Nihalani *et al.* 2011; Sujatha *et al.* 2012]. Initially, tools such as BASEBALL [Green Jr *et al.* 1961] were developed using the baseball league played in the US as a test case. BASEBALL provided answers related to location, dates, etc. This was followed by LUNAR [Woods 1973], a NLIDB system that was developed from the Apollo lunar exploration that provided information about soil samples. Other systems that followed were RENDEZVOUS [Codd 1974], LADDER [Sacerdoti 1977] and Chat-80 [Warren and Pereira 1982]. All these systems produced good results, but had a limited repository of information related to other domains [Mishra and Jain 2016; Papantoniou and Tzitzikas 2019]. Similarly, these systems used *hard-wired* knowledge and were dependent upon a limited application area, which was a major disadvantage. This has been improved by newer NLIDBs that cater for domain independence and multiple different databases. NLIDBs are classified into different categories. These categories show the knowledge-base used by NLIDBs generate answers to natural language questions [Affolter *et al.* 2019; Pazos R *et al.* 2013]. This section presents these categories of NLIDBs.

3.7.6.1 *Keyword-based*

The goal of keyword-based systems is to match keywords against a meta-data [Shah *et al.* 2013]. In this approach, the systems attempt to retrieve keywords from an input sentence and convert the equivalent into SQL queries. The following are some of the recent keyword-based tools:

SINA Shekarpour *et al.* [2015] developed `SINA` as an online, scalable keyword search system used for transforming natural language questions into SPARQL Protocol and RDF Query Language (SPARQL) queries. In its engine, it uses the HMM to establish the most likely NLQ from different datasets. To reduce questions into keywords, `SINA` uses the tokenisation, lemmatisation and stop word removal methods, which are segmented and processed, before the generated SPARQL query is displayed to the user. A major weakness of `SINA` is that it reduces the number of answerable questions because it translates only to conjunctive SPARQL queries.

AQQU A template-based system called `Aqqu` was proposed by Bast and Haussmann [2015] that uses keywords to translate natural language questions to their matching SPARQL query. The tool uses POS-tagging for entity matching, generates the set of sequences of words, computes the list of all entities from the knowledge base and provide scores for each of the entities matched. `Aqqu`'s strength lies in the identification of relationships between entities.

DEEPEYE The `DeepEye` tool was created by Qin *et al.* [2018] that attempts to use keywords to create visualisations from user's queries in natural language. The idea behind this tool is to use a keyword query and a dataset, to generate all possible visualisation. The study described the framework of `DeepEye` and explained that it crawls, stores and provides good visualisation from multiple different sources.

SPATIALNLI Li *et al.* [2019] presented a NLI tool called `SpatialNLI` that translates a natural language question into a structured SQL query that is executable by a DBMS. This tool learns from the keywords of the spatial comprehension model that takes a NLQ and captures the spatial-specific semantics of it. In addition, the tool uses a sequence-to-sequence approach, which is a deep learning concept to capture semantic meaning of a question. The accuracy reported by the tool indicates 90.7%. The authors claimed that `SpatialNLI` outperforms other popular state-of-the-art methods.

3.7.6.2 *Pattern-based*

The pattern-based systems are an extension of the keyword-based approach [Affolter *et al.* 2019]. This approach answers more complex questions and uses patterns to generate SQL queries from a natural language. Some of the pattern-based NLIDB systems are:

DBPAL Utama *et al.* [2018] developed `DBPal` as a novel data exploration tool that leverages advances in the deep model to provide a robust platform. This platform uses a pattern-based, deep learning model to translate NLQ into SQL queries. The tool attempts to support users in phrasing questions without knowing query features and database schema. `DBPal` uses the Paraphrase Database (PPDB) database that contains millions of paraphrases specified in 16 languages and uses this as an index that maps words to paraphrases for any given NL query.

NLQ/A The `NLQ/A` tool was designed as an NLI interface to query a knowledge graph [Zheng *et al.* 2017]. This tool was developed for end-users struggling to understand query languages such as SPARQL and SQL. In addition, it conducted experiments over the QALD dataset and showed that the approach was effective as it surpasses previous state-of-the-art tools with regards to recall and precision.

FANDA The `FANDA` was developed by Liu *et al.* [2019] as a NLI that uses a FollowUp dataset to generate a SQL query. The tool was targeted for novices struggling with SQL query, and intends to allow them write their request in a free-form specified in a natural language. `FANDA` employs a ranking approach specified in sentence patterns with a weakly supervised learning method to transfer across multiple different domains.

3.7.6.3 *Parsing-based*

In parsing-based systems, the tools parse the input natural language question and use the information structure to generate a query [Affolter *et al.* 2019]. In most cases, they use a dependency parser to handle any glitches. These systems use more advanced heuristics compared to the keyword-based and pattern-based approaches.

ATHENA Saha *et al.* [2016] developed an ontolog-driven system called ATHENA that enables users to write queries in natural language, which is then translated into a SQL query. ATHENA uses a two-stage approach where the input NLQ is first translated into an ontology, which is then translated into SQL. With ATHENA, the user is not expected to know how to write a query language such as SQL. The study concluded that ATHENA was used on three different open-source databases, and attained an impressive precision over them.

NALIR The NALIR system translates a correct English language sentence into a SQL query and evaluates the query against a RDBMS [Li and Jagadish 2014b]. The system consists of three parts: dependency parser for understanding the NL query linguistically, parse tree node mapper for node identification in the parse tree, and an interactive communicator that explains how the queries are to be processed.

BIOSMART Jamil [2017] presented the BioSmart tool that uses a syntactic classification that computes natural language sentence into several classes that fits into predefined syntactic templates, which is then interpreted to generate a SQL query. The generative process takes a natural language sentence by its NLP interface and maps this into a predefined sentence or query template. Next, the query mapper transforms the query into a logical query for the structural ontology which identifies the table, analysis tool and generates the query.

BELA The BELA tool was designed by Walter *et al.* [2012] as a QA system that processes natural language questions over linked data to generate a SPARQL query. Similarly, this parses the natural language questions and produces a set of query templates before generating the query. When compared to other systems, BELA attempts to reduce computation time and increases its user-friendliness for end-users to write correct SPARQL queries easily.

MANALA Giordani [2008] developed `MaNaLa` as a novel approach that exploits database meta-data and semantically maps natural language into SQL. The study showed that `MaNaLa` uses a machine learning algorithm that maps a dataset of natural language questions and SQL queries by their syntactic structures.

3.7.6.4 *Grammar-based*

Grammar-based systems use a set of rules represented as *grammar* that defines how the natural language questions can be used to generate a SQL query[Affolter *et al.* 2019; Song *et al.* 2015]. This supports end-users who are less knowledgeable about SQL to enable them to write correct queries. This section highlights a few grammar-based systems.

ASKNOW Dubey *et al.* [2016] developed the `Asknow` system where users can write their queries in English to a target database engine. The questions are first normalised into syntactic forms, before they are translated into SPARQL queries. In addition, the system is sufficiently adapted to query paraphrasing that enables it to use its grammar for the normalisation of a query which follows a syntactic process.

GFMED The `GFMed` NLI system was proposed by Marginean [2017] for biomedical linked data that applies a grammatical framework (GF)[12] that translates natural language queries into SPARQL queries. To generate natural language into SPARQL, `GFMed` uses GF for syntactic and morphological processes and aggregates the path before using RE operators for recognition. The study highlighted that `GFMed` can cater for a separate language apart from English for users to ask questions.

SQLIZER Yaghmazadeh *et al.* [2017a] presented `SQLizer` that uses a CFG for semantic parsing of natural language questions into SQL queries. `SQLizer` is an end-to-end system, which is fully automated to work with any database without needing additional customisation from the end-user. Due to natural language ambiguities, `SQLizer` adopts type-directed synthesis and repair techniques to generate a query, which is fully automated and non-database agnostic. In addition, `SQLizer` performed better when compared to the SQLizer system during an evaluation with three databases such as MAS [Sinha *et al.* 2015], IMDB [Lu *et al.* 2012] and YELP [Huang *et al.* 2014b]. Similarly, these datasets were used to evaluate NALIR [Li and Jagadish 2014a].

LN2SQL The `ln2SQL` tool was initially designed for another engine called `fr2SQL` to convert natural language in French into SQL [Couderc and Ferrero 2015]. In addition, this Python-based tool considers only the SELECT query command to alter a database using the French language. To parse the NL, `ln2SQL` uses a *treetagger* according to the Parts of Speech (PoS) tagging approach to filter words in a sentence. This treetagger is based on the spaCy NLP framework. The filtered words are then extracted and mapped into keywords which are used to generate a query. The generated query is used to retrieve rows from a table in a database. The study showed that `ln2SQL` can be used to support multiple databases.

3.7.6.5 *Speech-based*

Recently, speech-based NLIDBs have been introduced to ease conversations between humans and db applications without needing to typeset a request in natural language [Serban *et al.* 2016a]. A popular example is the Microsoft's Cortana [López *et al.* 2017; Hoy 2018] In this section, we present a few of these tools.

ECHOQUERY Lyons *et al.* [2016] built `EchoQuery` as a conversational system that uses speech commands to query a database system. The study concluded that `EchoQuery` was easy and flexible to use. Furthermore, the study was evaluated in Utama *et al.* [2017]. The evaluation was conducted using two baselines, such as template-based and rule-based approach to map a semantic tree to a SQL. The result of the evaluation showed that `EchoQuery` performed at more accurately when compared with two exisiting NLIDBs.

SPEAKQL Chandarana *et al.* [2017] presented an end-to-end speech driven interface used to convert NL into SQL queries. The authors combined four approaches to build the `SpeakQL` engine. The first of these processes was a state-of-the-art Automated Speech Recognition (ASR) technology that processes spoken SQL query into a transcribed output. Second, a transcribed output is processed to obtain a syntactically correct SQL statement that considers keywords, characters and literals using a CFG. Third, the literals are mapped to attribute names and values, then, a visual output is presented to the user. The study concluded that `SpeakQL` is friendlier, more interactive and significantly faster than other NLIDBs.

12 https://www.grammaticalframework.org/

CYRUS Godinez and Jamil [2018] developed a mobile speech assistant for the iOS[13] platform that supports large query classes on a test database. To parse a natural language into SQL, `Cyrus` uses a speech recogniser to convert a user's speech into text transcription. Also, it uses a *linguistic* tagger to parse the texts. In addition, it allows students to choose a sample database, perform operations on this database, and produce a result. The study showed that `Cyrus` was able to map simple NL statements into SQL successfully, and plans to improve its engine to accommodate complex queries.

3.8 FORMAL LANGUAGE AND AUTOMATA APPLICATIONS

In this section, we present some of the applications of FLA. First, we highlight some tools that have used REs, then we discuss CFG applications, and finally, we highlight a number of JFA applications.

3.8.1 *REs Applications*

This section contains some of the tools that have used REs.

INFORMATION RETRIEVAL Li *et al.* [2008] used REs for an algorithm called `ReLIE` to retrieve specific information from a corpus. The study compared `ReLIE` with a machine-learning extractor known as Conditional Random Field (CRF) algorithm. The study showed that ReLIE was more effective at extraction tasks compared to its counterparts.

CLINICAL APPLICATIONS A software tool by Turchin *et al.* [2006] was designed to identify and extract blood pressure values from clinical records. The study concluded that REs provide an alternative approach for abstracting data elements in multiple clinical applications if a general purpose NLP software is not available. A similar study used REs to find patterns in genes [Sharmila and Sakthi 2018].

COMPUTER SCIENCE EDUCATION `NOPRON` was designed by Ade-Ibijola *et al.* [2014] to aid the comprehension of novice programs. This tool translates these programs into detailed textual descriptions using REs.

SECURITY APPLICATIONS Xie *et al.* [2008] designed a framework called `AutoRE`, that detects spam mail using REs. The study concluded that this approach significantly reduced the false positive rate in the result. The study was extended to a tool named `BotGraph` [Zhao *et al.* 2009], developed to detect botnet spamming targeted at most email providers.

3.8.2 *CFGs Applications*

CFGs have been applied to many diverse tools in different domains. In this section, we present some of the tools that have used CFGs.

PROGRAM SYNTHESISER A program synthesiser that uses CFGs to automatically generate procedural programs in Python was developed by Ade-Ibijola [2018a]. The study emphasised that CFGs can be extended to automatically generate programs in other procedural programming languages.

FINANCIAL CHAT ANALYSER A tool developed by Ade-Ibijola [2016a] that uses CFGs for the automatic comprehension and summarisation of financial chats retrieved from the Instant Bloomberg messaging application.

13 A mobile operating system developed by Apple Inc

FUZZY SYSTEM A tool using CFGs to improve the elicitation of linguistic information in decision making was designed by Rodríguez *et al.* [2016]. This study showed that CFGs were used to provide a formal approach to the building linguistic expressions in fuzzy systems.

LYRICS GENERATOR Designed by Pudaruth *et al.* [2014], this uses CFGs rules and statistical constraints to automatically generate song lyrics.

PROFILE SYNTHESISER A tool that uses a variation of CFGs in the automatic generation of hypothetical social media profiles [Ade-Ibijola 2017c]. The study concluded that CFGs might be extended to similar problems in the health and social media domains.

3.8.3 *JFA Applications*

A number of studies have considered JFA for natural language abstraction. Here, we present a few of these studies.

AUTOMATA-LIKE SYSTEMS Cienciala *et al.* [2014] introduced Automaton-like P Colonies systems (or APCol systems) as formal methods that include JFA in membrane and distributed systems.

GAME THEORY Maubert and Pinchinat [2013] proposed the use of JFA for uniform strategies in game theory applications. Bozzelli *et al.* [2015] extended the study for winning conditions and module-checking scenarios using a JFA.

FREQUENTLY ASKED QUESTIONS Okwunma [2018] proposed the use of JFA for the abstraction of FAQs[14] in natural language for information retrieval tasks. The study extended this approach into a QA system (chatbot) that allows the comprehension of customer queries.

TWEET COMPREHENSION Obare *et al.* [2019] presented a tool called an Automata-Aided Tweet Comprehension (ATC) using a JFA for the automatic comprehension of tweets. The study reported that JFA was effective for this task.

3.9 THE GAP

This section highlights the outstanding areas of research, given the background of works that have been investigated. While there are many studies that have proposed tools that aid the comprehension of SQL, there is still a persistent need for new/improved methods to:

1. recognise using a formal approach such as REs for SQL queries that may seem confusing for users,
2. seek to recognise nested SQL queries that exist in different forms cascaded with balanced parentheses using another formal approach using a CFG,
3. create a tool that implements the synthesis of speech to SQL query,
4. generate SQL queries using a visualiser that applies visual specifications to build the queries, and
5. develop an approach to parse natural language into SQL query using a JFA. NLP is an area of AI, which introduces a higher complexity when SQL queries are synthesised from natural language.

14 Frequently Asked Questions

3.10 CHAPTER SUMMARY

In this chapter, the background related to SQL was presented. First, the challenges faced when learning and writing SQL queries were presented. Second, a number of pedagogical patterns and learning approaches were presented. Next, different state-of-the-art tools for the teaching and learning of SQL were highlighted. Then, a review of NLP and related areas was conducted. The last section enumerated the gaps that motivated this research. Part ii introduces two of the methods using *narrations* that were proposed for this research.

Part II

SQL COMPREHENSION AND NARRATION

Teaching queries in plain English eliminating syntactic barriers, is the goal of achieving SQL comprehension. This approach to teaching was first proposed by Fincher [1999] and regarded as the SFA. To aid program comprehension, this approach to teaching was termed *narrations* by Ade-Ibijola *et al.* [2014]. This approach helps to abstract a program without first considering the syntax of the language. Since SQL have less control structures, we have extended this approach of teaching for SQL for the first time, ignoring the syntax that describes unique sets of rules and guidelines of the language. The technique described was used for the automatic generation of explanations for simple queries using REs. In addition, to aid the understanding of nested queries, a CFG was used to recognise nested queries in an attempt to improve comprehension for these types of queries.

This part comprises of two chapters. Chapter 4 presents a formal technique, using REs to recognise simple SQL queries and describes the tool designed. In Chapter 5, a CFG was described for the recognition of nested queries and a tool was designed for the automatic generation of narrations for nested queries.

4 COMPREHENDING AND NARRATING QUERIES USING REGULAR EXPRESSIONS

The previous chapter outlined the background related to this research. This chapter presents a formal approach using *REs* for the recognition of different SQL query constructs. This was built into a tool called `S-NAR`, that automatically translates the recognised SQL constructs into textual explanations (i.e. narrations). `S-NAR` was tested with 5000 queries and some performance results were presented.

4.1 INTRODUCTION

REs are powerful methods for text processing which have played significant roles for many CS applications [Gogte *et al.* 2016; Harden 2017; Ganty and Valero 2019]. These applications range from operating systems and search engines to text editors, etc. In many programming languages, REs are inbuilt as standard libraries and present in the syntax of others [Cappers and van Wijk 2017]. Newer applications of REs are found in Logstash [Hamilton *et al.* 2018], Elasticsearch [Zamfir *et al.* 2019] and Grep [Ganty and Valero 2019] for finding patterns in texts. In this study, we have used REs for the recognition of SQL constructs. The formal definition of REs have been provided in Chapter 2.

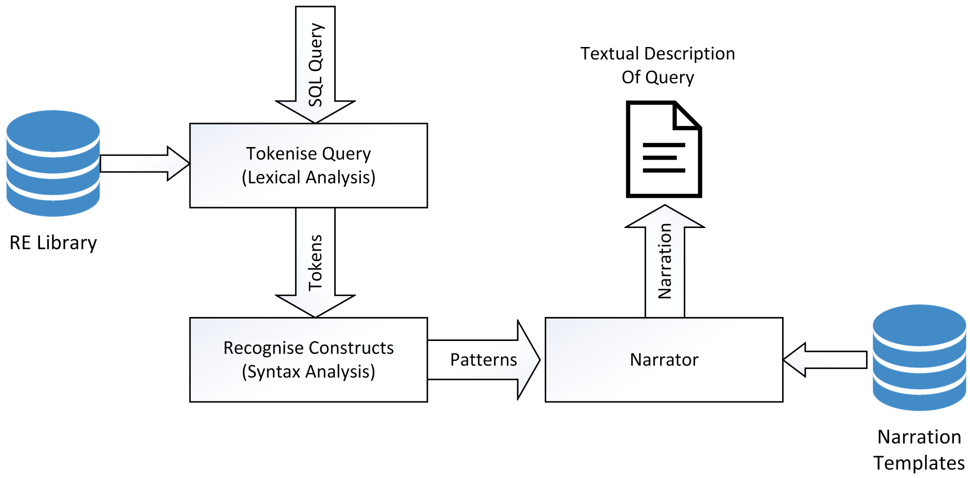


Figure 22: SQL query narration process

In this chapter, we propose a way to aid SQL comprehension via the automatic generation of narrations. Narrations are textual descriptions of queries that are presented in plain English, eliminating high level language and syntactic barriers often faced by novices. This style of teaching high-level languages was termed the SFA by Fincher [1999] and has been shown to aid comprehension of novice programs [Ade-Ibijola *et al.* 2014; Ade-Ibijola 2016b]. In this work, we have extended this approach to SQL queries for the first time and since SQL queries generally have lesser complex control structures (with no loops, nested constructs, etc), a recogniser based on the class of regular languages was used to parse and generate narrations from SQL queries. In Figure 22, we show how this technique works on SQL queries.

The query is first tokenised[1]. The tokens are grouped into syntactic parts at a recognition stage. The recognised sentential forms are then passed to a narrator. The narrator contains pre-defined templates that convert patterns of query into textual descriptions. Finally, the resulting narration is displayed to the novice/learner.

4.2 TRANSLATING QUERIES INTO NARRATIONS

Narrations have been applied in program comprehension [Ade-Ibijola *et al.* 2014; Ade-Ibijola 2016b], with no application to scripting languages such as SQL. In this section we describe how we have translated SQL queries to narrations. Narrations (as previously generated from programs) are step-wise descriptions of programming instructions and often longer than programs they represent. Narrations are different from comments because they do not provide much semantics as often used in programs — and are referred to as syntax-free textual algorithms [Ade-Ibijola *et al.* 2014]. For SQL queries, we present a sample narration. In Listing 3, we show a simple SQL query for creating a database table with five fields, while Algorithm 1 shows its narration.

Listing 3: SQL query to create a table with five fields

```
CREATE TABLE student_record(
StudentID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);
```

Algorithm 1 SQL query to create a table with six fields

This query creates a table named student_record and declares StudentID as an integer, Lastname as an alphanumeric entry of at most 255 character, Firstname as an alphanumeric entry of at most 255 characters, Address as an alphanumeric entry of at most 255 characters and City as an alphanumeric entry of at most 255 characters.

In this work, we have developed a tool that takes SQL queries and generates narrations similar to Algorithm 1. The next section of this work presents REs for the recognition of the syntax of SQL, a stage before narration generation.

4.3 REGULAR EXPRESSIONS FOR SQL ABSTRACTION

This section presents REs for recognising the syntactic components of queries, from characters to tokens and the various distinguished commands. First, we introduce a hierarchical diagram in Figure 23 that describes the different categories of SQL statements. This helps us in structuring the granularity level of the REs to be designed to recognise the queries. In Figure 23, we show how SQL can be broadly broken down into two categories of statements — DDL and DML. The DDL contains five major types of commands that are used to define/redefine database structures (such as Tables, Views, etc.) in the memory, while the DML is used to perform record-changing operations on the data committed to already existing objects. The relation in Figure 23 is used in structuring the REs used in parsing queries prior

1 Here, *Normalisation* is assumed to be a sub-stage of tokenisation.

to narration composition.

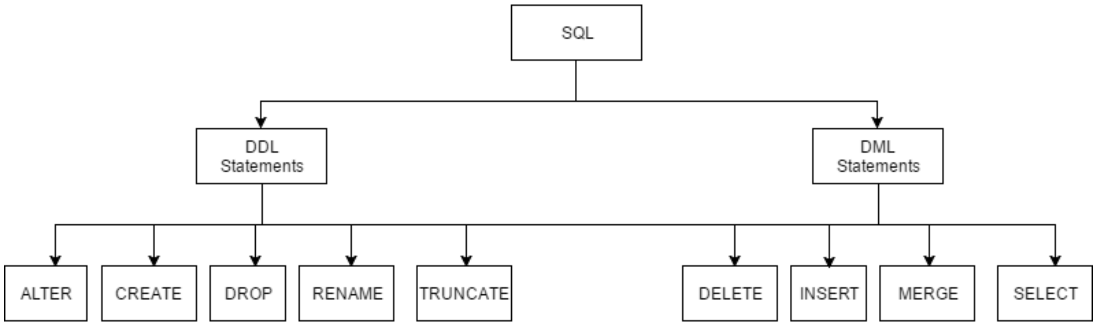


Figure 23: Categories of SQL commands

In Table 2, we present the basic lexemes in queries (across DML and DDL queries) that form a building block for the syntactic structures in higher levels of granularity. These include: specifications for identifiers (specified in similar fashion as in programming languages), the end of line delimiter, white spaces, commas, brackets, operators and values. Other streams of characters are also presented at this level such as: list of quoted and unquoted string/integer values that are separated by commas. These values often appear in many queries.

We proceed and present REs for DDL statements in Table 3. The ALTER construct allows for renaming a table or options to add, drop or modify columns of the table. Hence, we have two expressions in Table 3. The DROP RE recognises statements for deleting a database or a table. The RENAME, TRUNCATE and CREATE REs are similarly specified. The REs for recognising DML constructs are shown in Table 4, together with the different variations of the SELECT command.

4.4 INTRODUCING S-NAR

We have implemented `S-NAR` as a desktop application (Figure 24) using the regular expression library provided in the .NET framework [MSDN 2017]. We tested `S-NAR` with a dataset of 5000 SQL queries. `S-NAR` successfully narrated 4824 queries presented to it (about 96.48%). We noted that all failed instances were queries that had balanced parenthesis in them. This is because the language of balanced parenthesis is *nonregular*, hence, REs did not suffice in those instances. A parser based on a CFG will sufficient to handle this hitch. The remainder of this section shows and discusses some results from `S-NAR`.

4.4.1 *Implementation and Results*

In this section, we present results obtained from the narration of some queries during the testing stage of `S-NAR`.

4.4.2 *DDL Statements*

`S-NAR` was successful in narrating DDL statements. Listing 4 shows an ALTER statement and the generated narration for this statement is shown in Algorithm 2. `S-NAR` removes all the technical terms such as DROP, MODIFY, etc., and presents an intuitive summary. English words that are usually intuitive, such as ALTER, are left in the narration generated by `S-NAR`.

Table 2: Queries at the Lowest Granularity Level

<table>
<tr><th>Token</th><th>Abbreviation</th><th>RE (.Net)</th></tr>
<tr><td>Identifier</td><td>ident</td><td>[A-Za-z_][A-Za-z0-9_]*</td></tr>
<tr><td>Number</td><td>number</td><td>[1-9][0-9]*</td></tr>
<tr><td>Semi colon</td><td>semi_colon</td><td>\;</td></tr>
<tr><td>One or more white spaces</td><td>n_spc</td><td>\s+</td></tr>
<tr><td>Zero or more white spaces</td><td>spc</td><td>\s*</td></tr>
<tr><td>All</td><td>all</td><td>*</td></tr>
<tr><td>Assignment</td><td>ass_sym</td><td>\=</td></tr>
<tr><td>Bracket open</td><td>bra_open</td><td>\(</td></tr>
<tr><td>Bracket close</td><td>bra_close</td><td>\)</td></tr>
<tr><td>Comma</td><td>comma</td><td>\,</td></tr>
<tr><td>Greater than</td><td>greater_than</td><td>(\>)</td></tr>
<tr><td>Less than</td><td>less_than</td><td>(\<)</td></tr>
<tr><td>Not equal to</td><td>not_equal_to</td><td>(\!\=)</td></tr>
<tr><td>Not greater than</td><td>not_greater_than</td><td>(\!\>)</td></tr>
<tr><td>Not less than</td><td>not_less_than</td><td>(\!\<)</td></tr>
<tr><td>Greater than or equal to</td><td>greater_than_equal</td><td>(\>\=)</td></tr>
<tr><td>Less than or equal to</td><td>less_than_equal</td><td>(\<\=)</td></tr>
<tr><td>Logical operators</td><td>log_op</td><td>(AND|OR|ANY|LIKE|NOT|BETWEEN|EXISTS)</td></tr>
<tr><td>Arithmetic operators</td><td>ari_op</td><td>(\+|\-|*|\/|\%)</td></tr>
<tr><td>Comparison operators</td><td>comp_op</td><td>(\=|\!\=|\<|\>|\!\<|\!\>|\<\=|\>\=)</td></tr>
<tr><td>Integer value</td><td>int_val</td><td>(\-?\d+)[+-]?</td></tr>
<tr><td>Varchar value</td><td>varchar_val</td><td>(A-Za-z_)*</td></tr>
<tr><td>Boolean value</td><td>bool_val</td><td>(true|false)</td></tr>
<tr><td>Float value</td><td>flot_val</td><td>(\d\d?\.\d\d?)</td></tr>
<tr><td>Data type</td><td>datatype</td><td>(int|varchar|bool|float)</td></tr>
<tr><td>Ident separated by comma value</td><td>ident_sep_by_comma</td><td>((bra_open)(ident)(spc)(comma)(spc))* ((bra_open)(ident)(bra_close))</td></tr>
<tr><td>Value in quote</td><td>val_in_quote</td><td>((\')(bra_open)(ident)|(number) | (flot_val)|(n_spc)| (comma)(bra_close))+ (\'))</td></tr>
<tr><td>List of values in quote</td><td>list_of_vals_in_quote</td><td>(bra_open)(val_in_quote) (spc)(comma)(spc)(bra_close)* (val_in_quote))</td></tr>
<tr><td>List of values separated by comma</td><td>list_vals_sep_comma</td><td>(bra_open)(spc)(list_of_vals_in_quote) (spc)(bra_close)</td></tr>
<tr><td>Ident equals value</td><td>ident_equal_val</td><td>(ident)(comp_op)(val_in_quote)(comma) *(ident)(comp_op)(val_in_quote))</td></tr>
</table>

Table 3: DDL statement building blocks

<table>
<tr><th>Statement</th><th>Abbreviation</th><th>RE (.Net)</th></tr>
<tr><td>ALTER</td><td>alter</td><td>(ALTER)(_n_spc)(TABLE)(n_spc)(ident) (n_spc)(RENAME)(n_spc)(TO)(n_spc) (ident)(semi_colon)
(ALTER)(n_spc)(ident)(n_spc) (ADD|DROP|MODIFY)(n_spc)(COLUMN) (n_spc)(ident)(n_spc) (datatype)(semi_colon)</td></tr>
<tr><td>DROP</td><td>drop</td><td>(DROP)(n_spc)(DATABASE|TABLE)(n_spc)(IF) (n_spc)(EXISTS)(n_spc) (ident_sep_by_comma) (semi_colon)</td></tr>
<tr><td>RENAME</td><td>rename</td><td>(RENAME)(n_spc)(TABLE)(n_spc)(ident) (n_spc)(TO)(n_spc)(ident)(semi-colon)</td></tr>
<tr><td>TRUNCATE</td><td>truncate</td><td>(TRUNCATE)(n_spc)(TABLE)(n_spc)(ident) (semi-colon)</td></tr>
<tr><td>CREATE</td><td>create</td><td>(CREATE)(n_spc)(DATABASE|TABLE)(n_spc) (ident)(bra_open)(ident_sep_by_comma) (datatype)(semi_colon)</td></tr>
</table>

Table 4: DML statement building blocks

Statement	Abbreviation	RE (.Net)
DELETE	delete	(DELETE)(n_spc)(FROM)(n_spc)(ident)(n_spc)(WHERE)(n_spc)(ident)(comp_op)(val_in_quote)(semi_colon)
INSERT	insert	(INSERT)(n_spc)(INTO)(n_spc)(ident)(n_spc)(bra_open)(n_spc)(ident_sep_by_comma)(spc)(VALUES)(spc)(list_of_values_sep_by_comma)(semi_colon)
SELECT	select	(SELECT)(n_spc)(ident_sep_by_comma|all)(n_spc)(FROM)(n_spc) (ident)(semi_colon)
SELECT DISTINCT	distinct	(SELECT)(n_spc)(DISTINCT)(n_spc)(ident_sep_by_comma)(n_spc)(FROM)(n_spc)(ident)(semi_colon)
SELECT WHERE	where	(SELECT)(n_spc)(ident_sep_by_comma|all)(n_spc)(FROM)(n_spc)(ident)(n_spc)(WHERE)(n_spc)(ident)(comp_op)((\')(ident)(\')|(number))(n_spc)(AND|OR)(n_spc)(ident)(comp_op)((\')(ident)(\')|(number))(semi_colon)
SELECT WHERE_IN	where_in	(SELECT)(n_spc)(ident_sep_by_comma|all)(n_spc)(FROM)(n_spc)(ident)(n_spc)(WHERE)(n_spc)(ident)(n_spc)(IN)(n_spc)(list_of_values_sep_by_comma)(semi_colon)
SELECT WHERE_BETWEEN	where_between	(SELECT)(n_spc)(ident_sep_by_comma|all)(n_spc)(FROM)(n_spc)(ident)(n_spc)(WHERE)(n_spc)(ident)(n_spc)(BETWEEN)(n_spc)((\')(ident)(\')|(number))(AND)(n_spc)((\')(ident)(\')|(number))(semi_colon)
SELECT WHERE_LIKE	where_like	(SELECT)(n_spc)(ident_sep_by_comma|all)(n_spc)(FROM)(n_spc)(ident)(n_spc)(WHERE)(n_spc)(ident)(n_spc)(LIKE)(n_spc)((\')(ident)(\')|(number))(semi_colon)
SELECT ORDER	order_by	(SELECT)(n_spc)(ident_sep_by_comma|all)(n_spc)(FROM)(n_spc)(ident)(n_spc)(ORDER BY)(n_spc)(ident)(n_spc)(ASC|DESC)(semi_colon)
SELECT GROUPBY	group_by	(SELECT)(n_spc)(ident_sep_by_comma|all)(n_spc)(FROM)(n_spc)(ident)(n_spc)(GROUP BY)(n_spc)(ident)(semi_colon)
SELECT HAVINGCOUNT	having_count	(SELECT)(n_spc)(COUNT)(ident_sep_by_comma)(n_spc)(FROM)(n_spc)(ident)(n_spc)(GROUP BY)(n_spc)(ident)(n_spc)(HAVING COUNT)(n_spc)(ident)(comp_op)((\')(ident)(\')|(number))(semi_colon)

Listing 4: ALTER statement query

```
ALTER TABLE supplier ADD COLUMN
supplier_name varchar(255);

ALTER TABLE supplier DROP COLUMN
supplier_name varchar(255);

ALTER TABLE supplier MODIFY COLUMN
supplier_name varchar(255);
```

The CREATE statement in Listing 5 is narrated in Algorithm 3. In this case, S-NAR abstracted the varchar data type, and referred to it as *alphanumeric* — in order to aid comprehension. Listing 6 is a DROP statement that is narrated in Algorithm 4 — here the DROP keyword is abstracted as ERASE for ease of understanding.

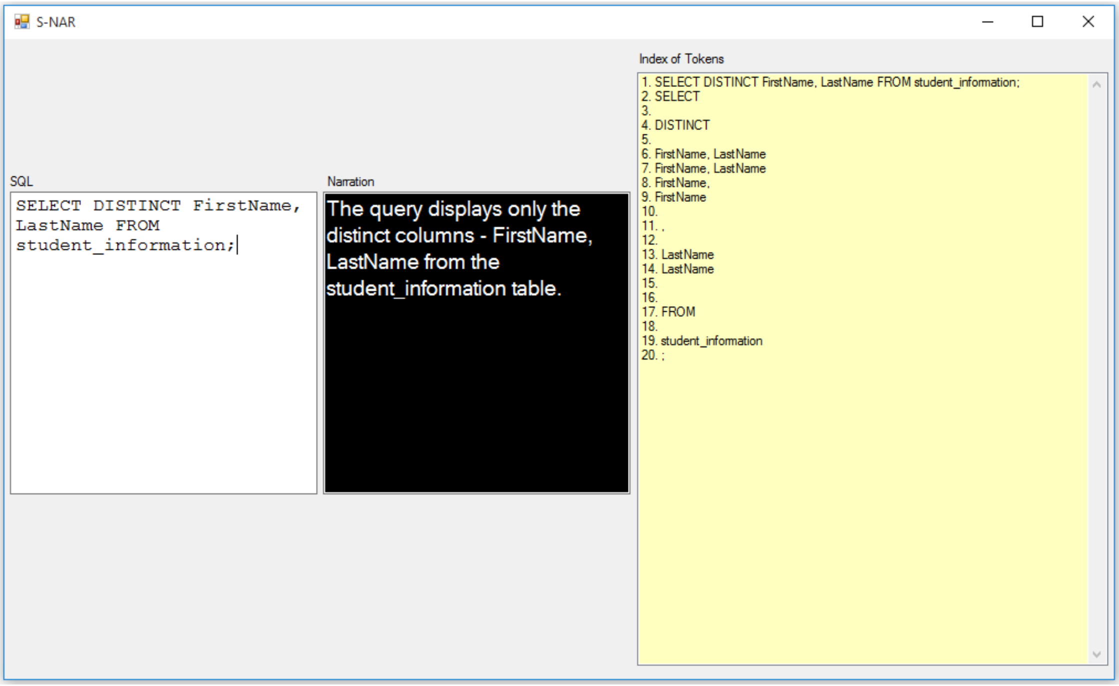


Figure 24: Screenshot of S-NAR during the narration of a SELECT query

Algorithm 2 Narration of the ALTER statement

1: *This query alters the supplier table by adding a new column called supplier_name that allows alphanumeric entry with at most 255 characters*
2: *This query alters the supplier table by removing a new column called supplier_name that allows alphanumeric entry with at most 255 characters*
3: *This query alters the supplier table by modifying a new column called supplier_name that allows alphanumeric entry with at most 255 characters*

Listing 5: CREATE statement query

```
CREATE DATABASE student_db;
CREATE TABLE student_information (
ID int,
Firstname varchar(255),
Lastname varchar(255),
Gender varchar(50),
Address varchar(255)
);
```

Algorithm 3 Narration of the CREATE statement

1: *This query creates a database named student_db*
2: *This query creates a table named student_information, and declares ID as an integer, Firstname as an alphanumeric entry of almost 255 character, Lastname as an alphanumeric entry of at most 255 characters, Gender as an alphanumeric entry of at most 255 character, Address as an alphanumeric entry of at most 255 characters.*

Listing 6: DROP statement query

```
DROP DATABASE IF EXISTS student_information;
```

Algorithm 4 Narration of the DROP statement

1: *This query erases the student_information database from the computer memory given that it previously exists*

Listing 7 and Listing 8 describe two queries for renaming and truncating a table. These queries are narrated in Algorithm 5 and Algorithm 6 respectively.

Listing 7: RENAME statement query

```
RENAME TABLE student_record TO student_information;
```

Algorithm 5 Narration of the RENAME statement

1: *This query renames the student_record table to student_information*

Listing 8: TRUNCATE statement query

```
TRUNCATE TABLE student_information;
```

Algorithm 6 Narration of the TRUNCATE statement

1: *This query empties the contents from the student_information table*

4.4.3 *DML Statements*

S-NAR was successful in narrating DML statements. Listing 9 shows the DELETE statement and its narrations. Here, the equals sign is abstracted to "is" as depicted in Algorithm 7. The INSERT statement in Listing 10 is narrated in Algorithm 8 with the keyword INSERT abstracted as ADD. The SELECT statement follows a similar narration pattern with its different variations (sometimes, having optional WHERE clause, DISTINCT and COUNT keywords, etc.) shown in Algorithm 9.

Listing 9: DELETE statement query

```
DELETE FROM student_information WHERE
student_firstname='peter';
```

Listing 10: INSERT statement query

```
INSERT INTO student_information (FirstName,
LastName, Address, City, PostalCode, Country)
VALUES ('Peter',
'Tom',
'21 claim street',
'Rivonia',
'2001',
'South Africa');
```

Algorithm 7 Narration of the DELETE statement

1: *This query removes from the student_information table where the student_firstname is peter*

Algorithm 8 Narration of the INSERT statement

1: *This query adds into the student_information table into columns; FirstName, LastName, Address, City, PostalCode, Country with details; Peter, Tom, 21 claim street, Rivonia, 2001, South Africa*

Listing 11: SELECT statement query

```
SELECT * FROM student_information;

SELECT DISTINCT FirstName, LastName
FROM student_information;

SELECT * FROM student_information
WHERE FirstName='peter' AND LastName='mark';

SELECT * FROM Customers
WHERE Country='South Africa'
OR City = 'Harare';

SELECT * FROM student_information
WHERE FirstName IN ('peter', 'john', 'felix');
```

Algorithm 9 Narration of the SELECT statement

1: *The query displays all information from the student_information table*
2: *The query displays only the distinct column - FirstName and LastName information from the student_information table*
3: *The query displays all the details from the student_information table where the FirstName is 'peter', and the LastName is 'mark'*
4: *The query displays all the information from the Customers table where the Country is 'South Africa', or City is Harare*
5: *The query displays all the information from the student_information table where the FirstName are either 'peter','john','felix'*

4.5 SCOPE AND LIMITATIONS

Up to this point, we have presented a tool called `S-NAR` that uses REs for recognising the constructs of simple DML and DDL queries, some of which had the `WHERE` clauses. Generating narrations for them was a relatively trivial task. One major limitation to note is that SQL queries sometimes come in more complex forms. For instance, it is possible to have queries and sub-queries, nested to several depths, and cascaded with *balanced parentheses*[2] as shown in Listing 12.

Here, there are two nested `SELECT` statements. The first one is a simple one that draws values from the Journals table, with the second part specifying the filter. Observe there are two pairs of parentheses in this query. Recognising this language will require more than REs;

2 The language of balanced parentheses is not a regular language.

i.e. a CFG for balanced parentheses can be used for this. This is similar to many programming languages, where REs are only useful for lexical analysis and not building parsers.

Listing 12: A non-regular nested SQL query

```
SELECT numcount,Jnls.*
FROM Journals J
WHERE numcount<=(SELECT COUNT(*/2)
                 FROM Journals);
```

4.6 CHAPTER SUMMARY

In this chapter, we have presented S-NAR, a software tool that translates SQL queries into textual description of the implied operations. S-NAR uses REs to first extract and group the tokens of an SQL query into syntactic categories and passes these grouped tokens to a module that uses predefined narration templates for the automatic generation of textual descriptions, referred to as narrations. S-NAR was also tested on 5000 queries scrapped from the Internet and it narrated a subset of these queries (96%) that do not contain balanced parentheses. This is only recognisable with CFGs or higher classes of formal abstract machines. We have argued that the generated narrations can aid the comprehension of SQL or be used to support teaching, in line with the SFA to programming language pedagogy.

Chapter 5 presents a CFG used to recognise nested queries cascaded with balanced parentheses. This handles the hitch faced with the current implementation of S-NAR.

5 COMPREHENDING AND NARRATING QUERIES USING CONTEXT-FREE GRAMMARS

In the previous chapter, a formal approach using REs was used for the recognition of SQL query constructs. A major limitation was identified, where REs were unable to recognise nested SQL queries. This chapter presents an extension to the use of *narrations* for describing nested queries cascaded with balanced parentheses using a *CFG*.

5.1 INTRODUCTION

CFGs are more powerful than regular languages (or REs) and have been used to describe nested parenthesis for programming languages [Cereda and Neto 2017; Bastani *et al.* 2017]. Hence, CFGs are formal notations used for expressing recursive definitions in programming languages. Since nested queries are defined recursively, this work extends the recognition of nested SQL queries using a set of rewriting rules (or production rules). To our knowledge, this appears to be the first time such an approach will be extended to recognise nested SQL queries. The formal definition of CFGs has been provided in Chapter 2.

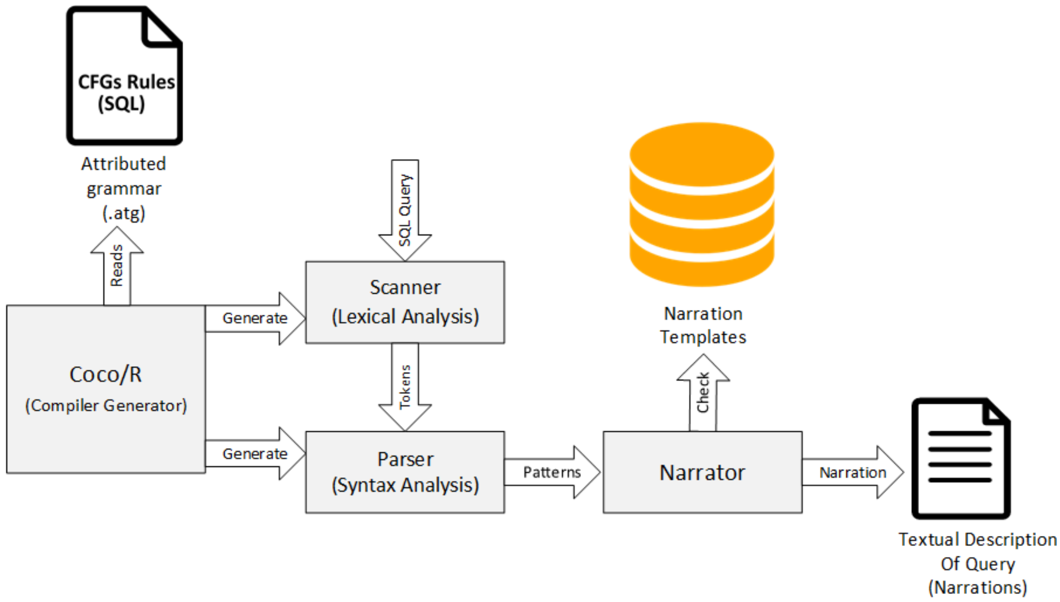


Figure 25: The framework of the SQL Narrator

This chapter presents an SQL Narrator that automatically generates narrations for a nested query. The term "narrations" was first coined by Ade-Ibijola [2016b] and was described as a textual description of programs in plain English and has been shown to aid the comprehension of novice programs [Ade-Ibijola *et al.* 2014; Ade-Ibijola 2017b]. The authors in [Ade-Ibijola and Obaido 2017] extended the use of narrations for describing simple queries using predefined templates. This was presented in Chapter 4. This chapter presents an improved version of the study that describes nested queries with balanced parentheses using CFGs – a subset of irregular languages to parse and generate narrations from SQL queries. Figure 25 shows how this approach generates a narration for a nested SQL query. First, the query is tokenised and grouped into a syntactic form for recognition, using Coco/R[1] that generates a scanner and parser. The sentential forms, derived as patterns, are then available to a narrator that checks these patterns for matches before converting the query into textual descriptions (or narrations). The generated narration is then presented to the learner.

5.2 CFG DESIGN FOR SQL QUERIES

This section presents the design of CFGs for the recognition of nested queries. This approach was used in the automatic generation of narrations for SQL queries. In this aspect, CFGs were designed using the compiler generator (Coco/R) that takes an attributed grammar, uses this grammar to generate a scanner and a parser. This is indexed in Appendix C. These generated elements (parser and scanner) were used to verify syntactic correctness of a query. To design the grammar, we adopt the Ron Savage's EBNF (Extended Backus–Naur Form) SQL grammar as described in [Savage 2017].

5.2.1 *Building Blocks*

To design the grammar G, we present the lexemes which are sequences of characters matched by a pattern for tokens used to build the production rules, P. These result in a set of terminal symbols Σ, such as letters, digits, num, etc (Productions 1 to 20). In Production 1 to 3, letters are defined and will appear in the list of identifiers (for example, student, lab01, etc). Productions 4 and 5 show numerical values that may appear. Productions 6 to 12 present semicolon, comma, open and close brackets, open and close quotes and period. Production 13 shows the symbol "all" which is used to display the entire information in a table/database. Production 14 shows whitespaces between strings.

$$\texttt{<letter>}^{*} \longrightarrow \texttt{<letter>(<letter>)}^{*} \tag{1}$$

$$\texttt{<letter>} \longrightarrow \texttt{A |...| Z| a |...| z} \tag{2}$$

$$\texttt{<ident>} \longrightarrow \texttt{<letter>}^{*}\texttt{<num>} \tag{3}$$

$$\texttt{<num>} \longrightarrow \texttt{<digit><digit>}^{*} \tag{4}$$

$$\texttt{<digit>} \longrightarrow \texttt{0 |...| 9} \tag{5}$$

$$\texttt{<semi\_c>} \longrightarrow \texttt{;} \tag{6}$$

$$\texttt{<comma>} \longrightarrow \texttt{,} \tag{7}$$

$$\texttt{<brac\_open>} \longrightarrow \texttt{(} \tag{8}$$

$$\texttt{<brac\_close>} \longrightarrow \texttt{)} \tag{9}$$

$$\texttt{<open\_q>} \longrightarrow \texttt{`} \tag{10}$$

$$\texttt{<close\_q>} \longrightarrow \texttt{'} \tag{11}$$

$$\texttt{<period>} \longrightarrow \texttt{.} \tag{12}$$

$$\texttt{<all>} \longrightarrow \texttt{*} \tag{13}$$

Productions 15 to 16 show the supported data type. Productions 18 to 20 present the operators such as arithmetic, relational and logical.

1 http://www.ssw.uni-linz.ac.at/Coco/

```
<wspace> ⟶ ws                                                      (14)
<type> ⟶ int | varchar | boolean | float                           (15)
<boolean> ⟶ true | false                                           (16)
<float_v> ⟶ <digit><digit><period><digit><digit>                   (17)
<arith_opr> ⟶ + | - | * | / | %                                    (18)
<rel_opr> ⟶ < | > | = | != | !< | !> | >= | <=                     (19)
<log_opr> ⟶ OR | XOR | AND | ANY | LIKE | NOT | EXISTS |
            BETWEEN | IN | IS NULL | UNIQUE                         (20)
```

We proceed to build productions for terms, identifiers and expressions used in Productions 21 to 25. Production 22 is defined recursively allowing the occurrence of either identifiers, numbers or decimal values.

```
<ident_sep_by_comma> ⟶ <ident_sep_by_comma>(<ident><comma>)*
                        <ident> | <all>                            (21)
<vals_in_qts> ⟶ <open_q>(<vals_in_qts><ident>|
                 <vals_in_qts><num>|<vals_in_qts>
                 <float_v>)*<close_q>                               (22)
<vals_list_qts> ⟶ <brac_open><vals_in_qts>
                   <brac_close>                                     (23)
<vals_list_sep_by_comma> ⟶ <brac_open><vals_list_qts>
                            <brac_close>                            (24)
<ident_condt_val> ⟶ <ident><rel_opr><vals_in_qts>                  (25)
```

We have presented the productions for the elements of our grammar as specified from Production 1 to Production 25. Next, we provide productions for the SELECT statement (<select_sub>) as seen in Production 26 and Production 29. The <select_sub> symbol satisfies the `SELECT...WHERE...FROM` statement and will be used to build the nested queries.

```
<select_stm> ⟶ <select_stm><select_sub><semi_c>                    (26)
<select_sub> ⟶ SELECT<wspace><cols_list><wspace>FROM<wspace>
               <ident><wspace>WHERE<wspace><condt_stmt>             (27)
<cols_list> ⟶ <cols_list>DISTINCT<wspace><ident>|
              <cols_list><ident_sep_by_comma>|
              <cols_list><ident>|
              <cols_list><all>                                      (28)
<condt_stmt> ⟶ <ident_condt_val>                                   (29)
```

5.2.2 *Nested SQL Queries*

A nested query is essentially a query inside another (**inner** and **outer**) query, which is common with the `SELECT` statements [Elhemali *et al.* 2007]. These statements are embedded within the `WHERE` or `HAVING` clause. In this section, we describe the productions for subqueries in the `UPDATE`, `DELETE`, `INSERT` and `SELECT` statements.

We start by defining the production for subqueries in the `UPDATE` statement. Production 30 describes the `UPDATE` subquery statement. The symbol <select_sub> allows the use of the `SELECT` query within the `UPDATE` statement to form a subquery. In most cases, this appears within the `IN` clause.

<update_sbqy> ⟶ UPDATE<wspace><ident><wspace>SET<wspace>
<ident><rel_opr><num><wspace>WHERE<wspace>
<ident><wspace>IN<brac_open><select_sub>
<brac_close><semi_c> (30)

Production 31 describes the DELETE subquery statement. This includes the <select_sub> symbol used to present the DELETE subquery statement.

<delete_sbqy> ⟶ DELETE<wspace>FROM<wspace><ident><wspace>WHERE
<wspace><ident><wspace>(<rel_opr>|<log_opr>)
<brac_open><select_sub><brac_close><semi_c> (31)

Production 32 defines the INSERT subquery statement. In this statement, the <select_sub> symbol is used within the INSERT query to describe the subquery.

<insert_sbqy> ⟶ INSERT<wspace>INTO<wspace><ident><brac_open>
(<ident_sep_by_comma>|<ident>)<brac_close>
<brac_open><select_sub><brac_close><semi_c> (32)

In Production 33, the SELECT subquery is described. In this subquery, the <select_sub> symbol is used within the WHERE clause.

<select_sbqy> ⟶ SELECT<wspace><cols_list><wspace>FROM<wspace>
<ident><wspace>WHERE<wspace><ident><wspace>IN
<wspace><brac_open><select_sub><brac_close>
<semi_c> (33)

In conclusion, we define the start symbol $S \in P$, used to begin the grammar G as described in Chapter 2. This is presented from Production 34 to 37.

<nested_qry> ⟶ <update_sbqy> | (34)
⟶ <delete_sbqy> | (35)
⟶ <insert_sbqy>| (36)
⟶ <select_sbqy> (37)

5.3 TRANSLATING NESTED QUERIES INTO NARRATIONS

In the previous section, the grammar design for nested queries was described. This section presents how nested queries are translated into textual narrations. Narrations are used to describe programs and they are generally termed *syntax-free textual* algorithms [Ade-Ibijola *et al.* 2014; Ade-Ibijola 2017b]. In Chapter 4, narrations were applied to describe simple SQL queries. The result showed that REs were not sufficient to generate narrations for complex queries. This section shows how CFGs are able to generate narrations for nested queries. In Algorithm 10, we show how the narration is generated. This recursive function takes a nested query (given as a list of queries and subqueries) and returns a concatenation of the query and all its subqueries.

For nested queries, we present the following narrations. The types of narrations we describe are **inner to outer** (flow from right to left), **outer to inner** (left to right) and **co-joined** (joining the queries together). Hence, the flow of information starts with the first query before ending with the query cascaded inside the balanced parentheses, and vice versa. In Listing 14, a row is deleted from a table using the nested query, and the corresponding narrations are presented in Algorithm 11, Algorithm 12 and Algorithm 13.

Algorithm 10 Generating Narrations

```
function getNarration(Q[i]) returns String          ▷ Q[i] = (q_1, q_2, ..., q_i)
    if (i = 1) then
        return Narrate(q_i)
    else
        return Narrate(q_i) + getNarration(Q[i − 1])    ▷ narration of Q
    end if
end function
```

Listing 13: Nested SQL query to delete a row

```sql
DELETE FROM country
WHERE city IN
(SELECT city
FROM country
WHERE city = "Pretoria");
```

Algorithm 11 Narration 1: Outer to Inner subquery

This query displays the city information from the country table where the city is equal to Pretoria and removes the entire information from the country table.

Algorithm 12 Narration 2: Inner to Outer subquery

This query removes the information from the country table where the city is contained in the values retrieved from the inner query, which gets all the city information that has a city equal to Pretoria

Algorithm 13 Narration 3: Co-joined subquery

This nested query contains two queries, where the first query removes the contents from the country table where the city appears in the second query which displays the city information from the country table where the city is equal to Pretoria

This example deletes from the *country* table a subset of rows whose *city* column value satisfies the condition specified in the `WHERE` clause. In this example, the `WHERE` ... `IN` clause specifies which rows to delete returned by the subquery. Hence, only the rows of the *country* table where the *city* is equal to “Pretoria" is displayed. The next section presents the implementation and results of the `SQL Narrator`.

5.4 IMPLEMENTATION AND RESULTS

The implementation of the `SQL Narrator` was carried out using the C# as the primary language that runs on the .NET Framework. This tool was tested with datasets of nested queries (in Appendix E) and successfully narrated them. An example of the narration using the `SQL Narrator` is shown in Figure 26.

To use the `SQL Narrator`, a user is expected to enter a nested query into the querybox. The nested query is then converted from characters into tokens and grouped into a sentential form before the narration is displayed to the user. A help file is available to the user. This help file contains a series of steps required to use the narrator.

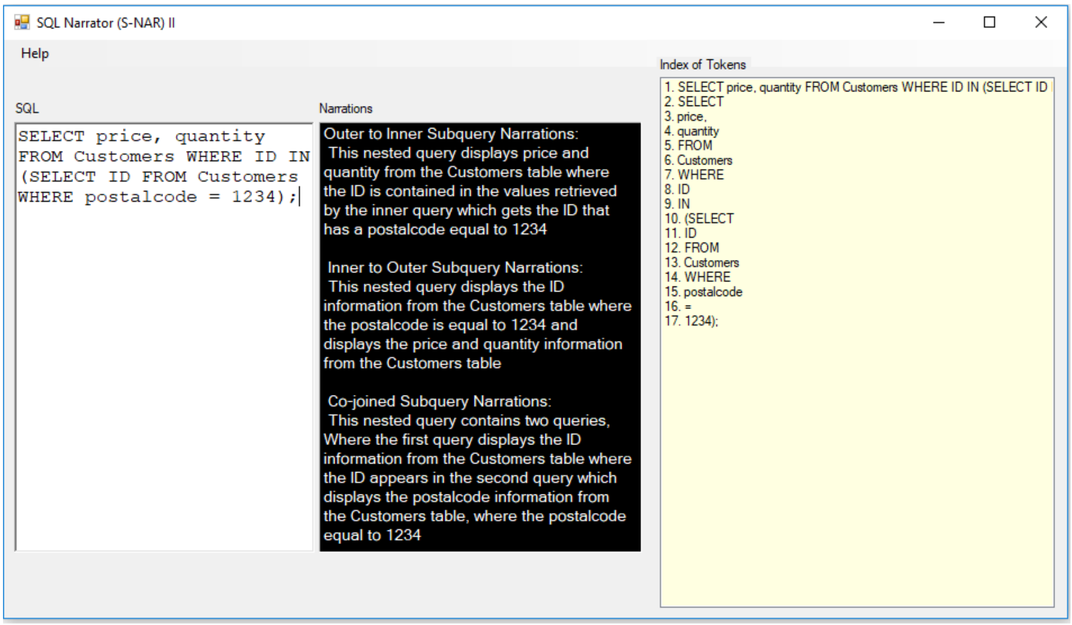


Figure 26: The process of narrating a nested query

The narration approach that has been presented in this work can be integrated into a SQL pedagogy to assist students in learning nested queries for the first time. We believe that this e-pedagogy will make it easier for students to understand nested queries.

5.5 CHAPTER SUMMARY

In previous studies within programming pedagogy, it was shown that narrations could be used to enhance learning and assist novices in comprehending programming languages. Another study examined the use of narrations for simple SQL queries. This work presents a new direction of using narrations to assist learner understanding of nested queries.

In this chapter, a grammar-based approach that automatically translates nested SQL queries into narrations was presented. This approach used the CFG formalism based on the Coco/R parser generator that takes an attributed grammar and generates a scanner and parser. This approach was implemented in a `SQL Narrator` based on the C# language that runs on the .Net Framework. The `SQL Narrator` was tested with a dataset of nested queries and the narrations were presented. In Part iii, we present the synthesis aspect of this study.

Part III

SQL SYNTHESIS

Automatic synthesis of problems into SQL is instrumental to end-users with diverse backgrounds such as business analysts, financial professionals, marketing personnel, etc [Wang *et al.* 2017a; Yaghmazadeh *et al.* 2017a]. These users frequently use db applications but lack the technical expertise to write a correct SQL query. Although these users can clearly describe what tasks they intend to perform, they are often faced with how to specify what the intended query should be. Such confusion may increase if they frequently need to engage with technical staff or seek help through online forums just to perform their daily operation. Such process can be time-consuming and frustrating. To mitigate these challenges, we propose different interactive user interfaces that these users can engage with. First, we present a tool that allows a user to specify their query requests in a free-form termed as *narrations*, then we show an interactive visualiser that uses drag and drop interactions to generate a query using icons. Last, we present a speech-query tool that takes a speech input and converts this into a query output.

This part contains three chapters. In Chapter 6, we describe the translation of narrations into SQL queries and in Chapter 7, the visualiser for SQL queries is presented. Chapter 8 describes the speech synthesiser tool for SQL queries.

6 NARRATIONS TO SQL QUERY

In Chapter 5 of Part ii, we presented a grammar-based approach that automatically translates nested SQL queries into narrations. This chapter introduces the synthesis aspect of this thesis. In this chapter, textual *narrations* depicted as natural language descriptions are translated into SQL queries and the resultant feedback is provided to the user. This approach uses a JFA, and was integrated into a tool called `Narrations-2-SQL`.

6.1 INTRODUCTION

JFA is an automata-based algorithm used for processing discontinuous information [Meduna and Zemek 2014; Meduna and Soukup 2017]. This algorithm has been used in many domains due to its expressive power [Meduna 2014; Fernau *et al.* 2015]. Since natural languages are highly ambiguous in nature, we have extended the use of JFA to synthesise SQL queries from natural language specifications. To our knowledge, this is expected to be the first time in which such an approach will be applied for SQL query translation from natural language. Our approach allows users to express their query in a *free-form* – in natural language to produce the equivalent SQL query [Li and Jagadish 2014b; Norouzifard *et al.* 2008]. It should be noted that this free-form approach expressed in natural language is regarded as *narrations*, and has shown to improve program comprehension [Ade-Ibijola *et al.* 2014; Ade-Ibijola 2016b]. The formal definition of a JFA has been provided in Chapter 2.

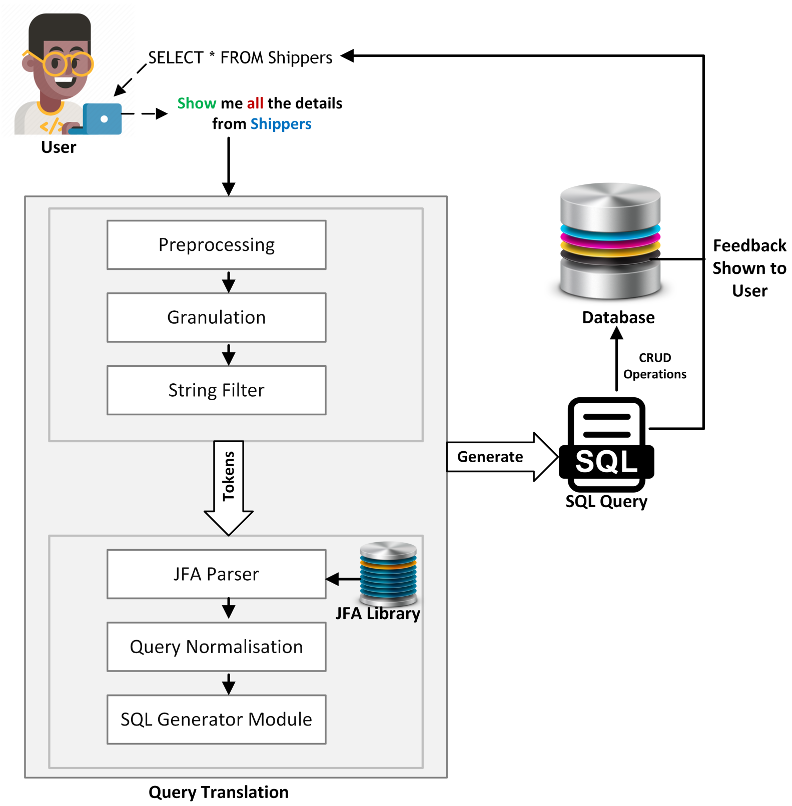


Figure 27: The framework of `Narrations-2-SQL`

This chapter presents the use of natural language specifications in a tool called `Narrations-2-SQL`, to aid the understanding of SQL. The `Narrations-2-SQL` engine uses a JFA – a type of Finite Machine, to translate natural language specifications into SQL queries, executes the query and provides feedback to a user. Figure 27 shows the framework of `Narrations-2-SQL`. A user typesets a NLQ, which is then processed by the `Narrations-2-SQL`. To begin, `Narrations-2-SQL` preprocesses the typed texts into a stream of tokens, which is then passed to the JFA for matches. At this phase, the tokens are matched and used to construct an SQL query. The equivalent query is used on the `XNorthwind DB` [Ade-Ibijola and Obaido 2019] and the feedback is provided to the user.

6.2 NATURAL LANGUAGE TO SQL QUERY

In this section, we show how we have abstracted natural language to a JFA. Our aim at this phase is to remove unwanted features from the dataset and keep the relevant details (tokens) as shown in Figure 27. To abstract the tokens into a JFA, we use the states, transition process and data representation to understand the abstraction process. The `XNorthwind` database consists of eight tables and 100000 iterations of datasets, which was used to train our JFA.

6.2.1 *JFA Design*

To abstract natural language specifications into a JFA, we identify the entities with matching colours that make up the alphabet. These entities are query type or ($\sum_{QT} = a_x$) in *green*, column type or $\sum_{CT} = b_y$ in *red* and entity or $\sum_{ET} = c_z$ in *blue*. Table 5 shows the JFA symbols for the XNorthwind database used for this work. We show an example of a JFA and the language it accepts.

Given a typed request from a user:

1. Please, help me to find all the employees' information that work for this organisation

The JFA that follows:

$$\mathbf{M} = (\{R, S, T, U\}, \{a_5, b_{21}, c_7\}, R, R; \{U\})$$

$\{R, S, T, U\}$ are the states,
$\{a_5, b_{21}, c_7\}$, are the input alphabets,
R is the set of rules.
R is a start state,
$\{U\}$ is a final state.

with

$$\mathbf{R} = \{Ra_5 \rightarrow S, Sb_{21} \rightarrow T, Tc_7 \rightarrow U\}$$

accepts

$$L(M) = \{w \in \{a_5, b_{21}, c_7\}^{*}\colon |a_5| = |b_{21}| = |c_7|\}$$
$$\text{i.e. } a_5 = \mathtt{find}; b_{21} = \mathtt{all}; c_7 = \mathtt{employees}$$

$$\begin{aligned} b_{21}a_5c_7b_{21}c_7\underline{R}a_5 &\curvearrowright b_{21}a_5c_7\underline{S}b_{21}c_7 && [Ra_5 \rightarrow S] \\ &\curvearrowright b_{21}a_5c_7\underline{T}c_7 && [Sb_{21} \rightarrow T] \\ &\curvearrowright \underline{U}b_{21}a_5c_7 && [Tc_7 \rightarrow U] \end{aligned}$$

Figure 28 shows a JFA with its transitions. It has four states, labeled as R, S, T, U. Here, the start state is denoted as R and the accepting state, U, denoted by the double circle. From the diagram, Ra_5 moves to S, where the only string found is the *find* keyword. The second transition shows the movement of Sb_{21} to T showing only the *all* keyword. Last, Tc_7 moves

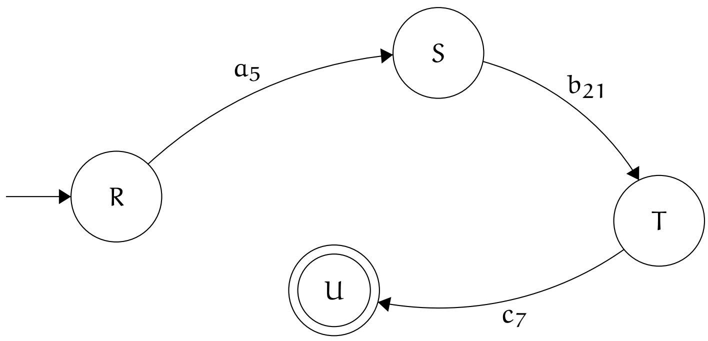


Figure 28: A JFA for Example 1

to U showing the *employee* keyword. It is worth noting that this example only shows four states with corresponding transitions. We can have as many states and transitions, depending on the input statement specified from the user.

Table 5: JFA symbols from 1 - 24

JFA symbols		
a_x	b_y	c_z
a_0 Display	b_0 OrderID	c_0 Suppliers
a_1 Select	b_1 CompanyName	c_1 Products
a_2 Show	b_2 ContactName	c_2 OrderDetails
a_3 List	b_3 Supplierscol	c_3 Orders
a_4 Return	b_4 ContactTitle	c_4 Categories
a_5 Find	b_5 Address	c_5 Employees
a_6 Compute	b_6 City	c_6 Customers
a_7 Get	b_7 Region	c_7 Shippers
a_8 Remove	b_8 PostalCode	–
a_9 Clear	b_9 Country	–
a_{10} Delete	b_{10} Phone	–
a_{11} Change	b_{11} Fax	–
a_{12} Update	b_{12} HomePage	–
a_{13} Add	b_{13} ProductID	–
a_{14} Give	b_{14} ShippersID	–
a_{15} Discontinue	b_{15} CategoryID	–
a_{16} Make	b_{16} Quantity	–
a_{17} Increase	b_{17} UnitsOnOrder	–
a_{18} Create	b_{18} ReorderLevel	–
a_{19} Read	b_{19} Discontinued	–
a_{20} Insert	b_{20} Productscol	–
–	b_{21} All	–
–	b_{22} ShipPostalCode	–
–	b_{23} HireDate	–
–	b_{24} Extension	–

6.2.2 *Query Normalisation*

This phase takes the keywords extracted from the JFA design to semantically form a proper sequence for the SQL generator module. This approach was further strengthened using WordNet[1]. For example, words such as select, choose, pick and display are mapped to SELECT keyword. Words such as insert, add, increase and build are mapped to the INSERT keyword. Also words such as update and amend are mapped to the UPDATE keyword. For the DELETE keyword, words such as remove and delete are mapped. The attributes of the XNorthwind tables are the column details. Once mapped, the information is fed to the query generator.

6.2.3 *SQL Generator Module*

The query generator transforms the semantic information at the normalisation phase and generates an SQL query. Since this work is limited for a single-relation, this phase is quite straightforward. The generated query is used against the XNorthwind DB and the result is displayed to the user.

6.3 IMPLEMENTATION, RESULTS AND APPLICATIONS

6.3.1 *Implementation and Results*

The JFA technique described in this chapter was implemented into a tool called `Narrations-2-SQL`. This tool was developed as a C# application specified using the Microsoft .NET framework. The implemented tool was tested with 204 crowdsourced queries specified in natural language (as presented in Appendix D), sourced from the `XNorthwind DB`. The `XNorthwind DB` was used in this study, which comprises eight tables and 100000 tuples. The result of the implementation can be found in Figure 29.

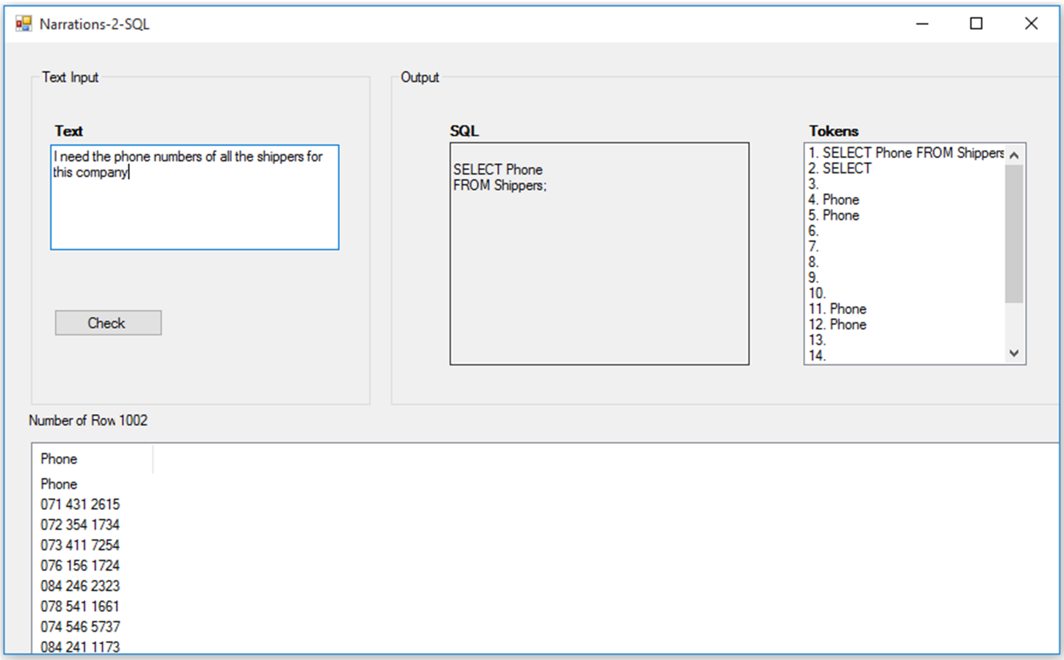


Figure 29: The user interface of `Narrations-2-SQL`

1 https://wordnet.princeton.edu/

6.3.2 *Applications of Narrations-2-SQL*

We present possible applications of the tool we have designed for this study. As a tool, `Narrations-2-SQL` can be used as a:

1. QA system,
2. teaching and learning aid,
3. tutoring system for SQL,
4. query tool for complex BI systems, and
5. natural language interface to query databases.

6.4 CHAPTER SUMMARY

This chapter describes a new approach to translating SQL queries from natural language. The technique described in this work uses a JFA, designed into a tool called `Narrations-2-SQL` for the purpose of SQL query translation from narrations. This work appears to be the first application of abstracting natural language specifications to a JFA, in addition to mapping this to an SQL query. This is a major contribution to the problems faced in the information retrieval (IR) domain. In its framework, `Narrations-2-SQL` performs operations on the `XNorthwind` database using simple SQL commands to create, retrieve, modify and delete data. Feedback of the operation is presented to a user. If implemented on a large scale, `Narrations-2-SQL` will assist end-users in different domains, to specify their queries in natural language, and perform their tasks seamlessly without needing much help from technical users. More so, our evaluation shows that the majority of the users agreed that this approach can be useful in industry.

Chapter 7 describes the use of visual specifications that explores *drag and drop* interactions of query-like images to generate SQL queries.

7

VISUAL SPECIFICATIONS TO SQL QUERY

In Chapter 6, we described the translation of natural language descriptions, termed *narrations*, into SQL. These narrations are English-like descriptions specified by end-users in natural language forms. This idea was implemented into a software tool called `Narrations-2-SQL`. The tool is expected to assist end-users in writing correct queries, which was a problem identified in the literature. This chapter presents a tool that uses pre-defined images that represent SQL commands to generate a query. This study is expected to improve students' comprehension of the SQL query concept.

7.1 INTRODUCTION

Visual specifications are symbols used to represent features, which can be used to display some text or program [Rojit *et al.* 2016]. In the programming concept, visual specifications have been used to build and demonstrate a programming solution especially with problems faced by students [Roberts *et al.* 2019; Eden *et al.* 2018]. This problem is not only limited to program understanding; students struggle to memorise database schema [Kawash 2014; Cembalo *et al.* 2011; Garner and Mariani 2015]. In addition, writing DML expressions has shown to be problematic for students Dekeyser *et al.* [2007]. In order to provide adequate support to address these problems, there is a need to build interactive platforms, incorporated with either *animation* or *visualisation* aids to support the understanding of SQL. In past decades, a number of tools have been developed to provide support in learning SQL [Cembalo *et al.* 2011; Folland 2016]. Some of the existing tools employed interactive visualisations to aid SQL understanding.

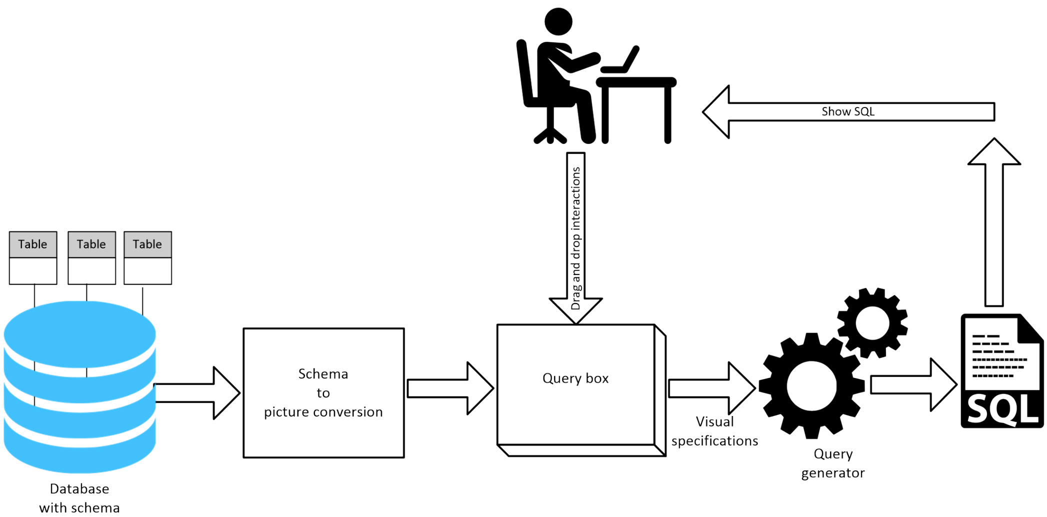


Figure 30: The framework of the SQL query generation

In this chapter, we propose the use of an interactive visualisation technique to aid the understanding of SQL. The visualisation technique ensures the interaction between visual specifications to build queries and will eliminate the need to memorise database schemas, which is a major problem faced by students learning SQL. In this work, we have developed an interactive visualisation aid called `SQL visualiser`, which uses visual specifications of the 'drag and drop' interactions for generating SQL queries. Although, this approach has found application in the programming languages paradigm such as `Alice` [Dann *et al.* 2008], `Scratch`

[Resnick *et al.* 2009] and `StarLogo TNG` [Wang *et al.* 2006], it has not (to the best of our knowledge) been applied to SQL queries. Figure 30 shows the SQL query generation process. To use the `SQL visualiser`, images are represented as visual specifications to depict the database model. These images can be moved into the query box. When the moved images correspond to a standard SQL `SELECT` statement, a query is generated and displayed to the user.

7.2 DESIGN OF THE SQL VISUALISER

One aspect of queries which poses difficulties for students is the SQL `SELECT` construct [Sadiq *et al.* 2004; Qian 2012]. This type of query is used to extract data from a relational database [Kearns *et al.* 1997]. Hence, our main goal is focused on using visual aids to easily generate queries by means of the SQL `SELECT` constructs. This idea can be extended to the system of queries.

It is a common perception that students are better at *recognising* visual constructs rather than at *writing* codes [Dekeyser *et al.* 2007]. Thus, this work is motivated by an intention to use visual specifications in order to generate SQL queries. Also, another motive of developing this SQL visualiser is to use it as a teaching and learning aid. We have found that database schemas pose difficulties for students [Dekeyser *et al.* 2007], hence, our intention is to simplify the process of understanding database schemas. We identify three main points to distinguish our visualisation from other approaches.

INTUITIVE Our visualisation is intuitive to students who are learning SQL queries for the first time. The visualisation uses images to depict each query statement. It helps students better understand SQL queries since it helps them get a glimpse of the behaviour of the image when each image is selected. Hence, students do not require extensive training to understand how to use the visual aid.

INTERACTIVE The visualisation tool is interactive, which means that students are not required to write any query statement in the application. They can simply click and drag the images across panels. Query statements are generated at the same time.

HELPFUL A help facility is provided before using the visualiser. A user is provided with an instruction of the underlying database schema before using the application. Also, *hints* are provided to the user and are specified using colours (green or red). These colours show whether a query is wrong or correct. In addition, a textual suggestion is offered to the user to ensure that the correct object is selected.

The `SQL visualiser` was implemented as a Windows Form Application and was included as part of the .NET framework for the purpose of creating rich client applications [Liberty 2005]. The visualisation tool consists of some components used for the generation of a query. These components include schema, query box and query generator.

7.2.1 *Schema*

The schema shows the logical organisation of the data. In the visualiser, the schema consists of tables and associated fields. The SQL query is based on the underlying schema. In addition, if a schema is correct, the generated SQL query is correct, and vice-versa. Figure 31 depicts the schema for this model.

In the schema, each table is shown by its name displayed at the top and its corresponding attributes shown at the bottom. For example:

Lecturer (id, name, phone, email, department)
Courses (id, title, credits, description)
Student (id, name, gender, age, address)

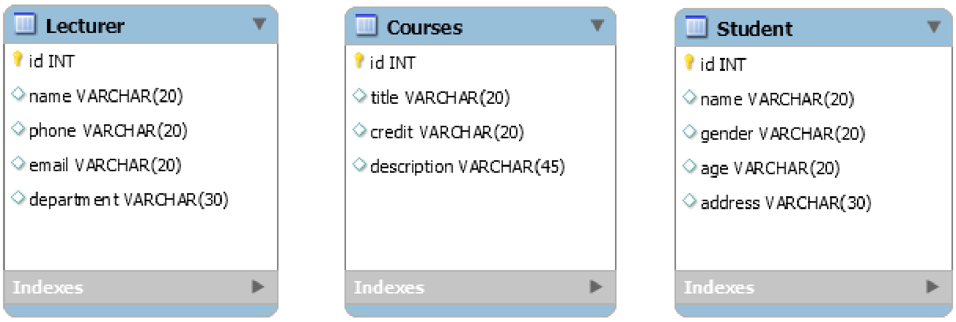


Figure 31: Logical organisation of the data

Each table is linked by a primary key. The unique identifier for the tables is the entity *id*. The visualisation tool relies on the schema to generate the SQL query.

7.2.2 *Query Box*

The query box is used to specify subsets of the schema that the user is interested in. The query box is also denoted as the building block for the query generator. In the query box, the schema (represented by the images) are extracted into a form used by the query generator to generate the SQL query. Each image is included with a caption for easy identification. Table 6, Table 7, Table 8 and Table 9 represent the pictures and descriptions used to represent the schema.

Table 6: Symbols and description of the SELECT statement

Symbol	SQL Block	Description
	`SELECT`	This icon represents the SELECT statement

Table 7: Symbols and descriptions of the Lecturer table

Symbol	SQL Block	Description
	`*`	This denotes "all" in rows
	`id`	This icon represents the primary key field
	`name`	This icon denotes the name field
	`phone`	A representation for a phone field
	`email`	This symbol denotes an email field
	`Lecturer`	A representation for a lecturer field

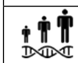
7.2.3 *Query Generator*

The query generator transforms the images in the query box and presents a query to the user. As more images are added, the query generator also adds the attribute to the query. The generator phase is very straightforward since the scope of this work is limited to a single-relation. The `SELECT` portion of the query consists of tables and attributes; where the `FROM`

Table 8: Symbols and descriptions of the Courses table

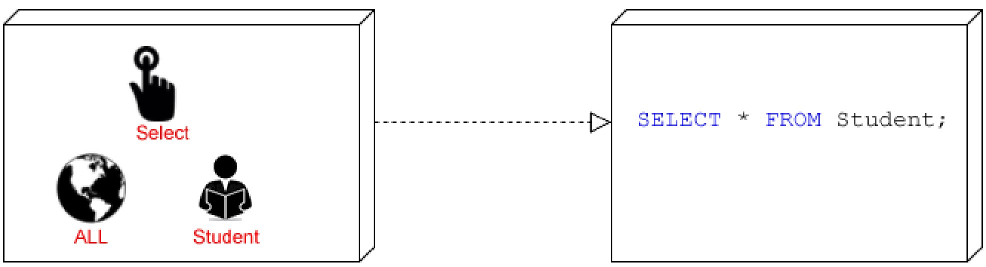
Symbol	SQL Block	Description
	`*`	This denotes "all" in row
	`id`	This icon represents a primary key field
	`title`	This icon denotes a title field
	`credit`	A representation for a credit field
	`description`	This symbol represents a description field
	`Courses`	A representation for a course entity

Table 9: Symbols and descriptions of the Student table

Symbol	SQL Block	Description
	`*`	This denotes "all" in rows
	`id`	This icon represents an identity field
	`name`	An illustration for a name field
	`gender`	A symbol for a gender field
	`age`	A representation for age field
	`address`	An icon for an address field
	`Student`	This symbol represents a student entity

clause defines a table and the `WHERE` clause is defined by a field attribute and its value. We illustrate this in Example 7.2.1.

Example 7.2.1. *Consider a simple database table with the schema: Student (id, name, age). Now, write a simple SQL query to display all information from the student table.*



(a) Query box (b) Query generator

Figure 32: The process of generating an SQL query

Figure 32 presents the process that the query generator uses to present queries. When a user adds the images into the query box (a), the query generator displays the visualisation from the images that were selected into the query box (b). More options used in this chapter are presented in the next section.

7.2.4 *More Options*

OPERATORS AND VALUES In the SQL visualiser, we have explored some comparison operators, such as the (= or *equal to*, $<$ or *less than* and $>$ or *greater than*). The comparison operators are used within the generated query with values between 0 – 100 to show the relationship. For each operator, the user can interactively determine which option to select. Further, a user can select the preferred choice of value to use within the query by using the scroll option provided. Once the scroll option is selected, it changes the value within the generated query.

COLOURS In the HCI interface design specifications, colours have been described to convey information [Brown 1998]. Within this specification, the association of colours may be used for many purposes if this is implemented conservatively. Colours have salient features, which are useful in human perception [Jost *et al.* 2005]. For example, the colour *red* strongly indicates an error, while *green* indicates a normal or acceptable condition. These colours were explored within the visualiser to indicate either an acceptable condition or to respond criticality to a user's error as presented in Figure 33.

Red (indicator of an error)	Green (acceptable condition)
Incorrect query, drag the required field	**Fantastic! Your query is correct**

Figure 33: Colours used to show annotations

7.3 RESULTS

We present the result from using the visualisation tool. Figure 34 shows the feedback received from adding only the `SELECT` operation into the query box at runtime. The feedback received will assist the user to specify the required visuals before the query can be generated. This example shows that if the user inserts the correct table and its attributes, the query will be successfully generated. The field "ID" was chosen as the primary key for each of the table, and the value, "50" was selected; See Figure 35. The help facility showing the instruction on how to use the `SQL visualiser` is presented in Figure 36. `SQL visualiser` was compared with a number of SQL visualisation tools. Table 10 shows the result of the review.

Table 10: Existing tools versus the SQL Visualiser

Features	**eSQL**	**SAVI**	**SQlify**	**sAccess**	**eledSQL**	**QueryViz**	**Our tool (SQL Visualiser)**
Visualisation of database schema	✗	✓	✓	✗	✗	✓	✓
Visualisation of output data	✓	✓	✓	✓	✓	✗	✗
SQL query generation	✗	✗	✗	✗	✗	✗	✓
Feedback on query semantics	✗	✗	✓	✗	✗	✓	✓
Visual object representation	✗	✗	✗	✗	✗	✗	✓
Ideal for less knowledgeable users (undergraduate students)	✗	✗	✗	✓	✓	✗	✓

We have presented a visualisation tool that applies visual specifications to generate queries. The technique presented in this chapter will find applications in teaching and learning systems. The benefits offered by the visualiser will facilitate human comprehension of SQL

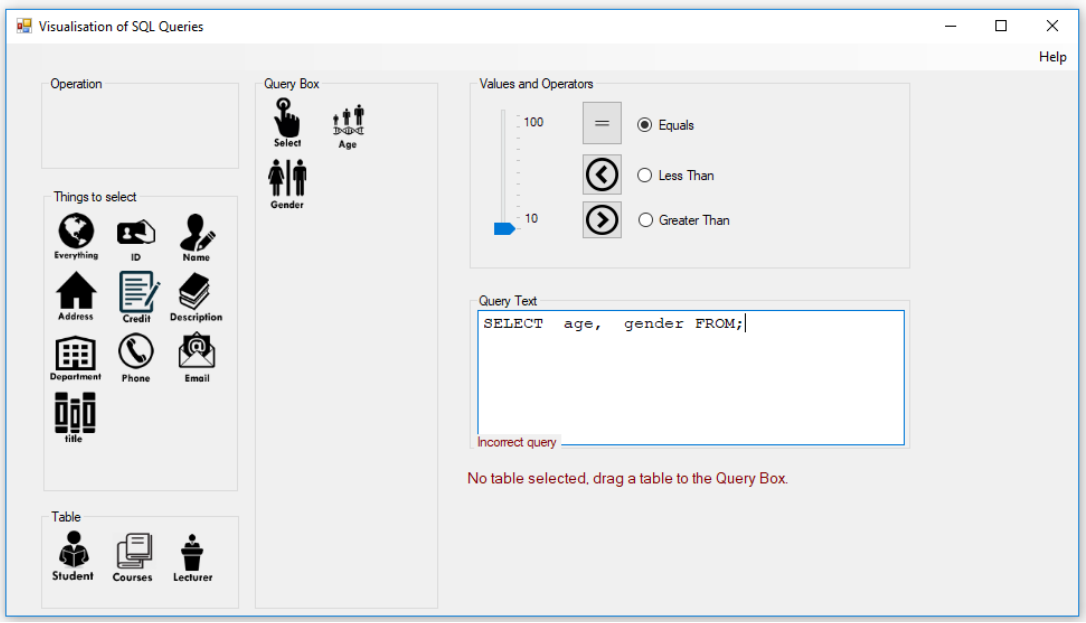


Figure 34: Hints provided to the user

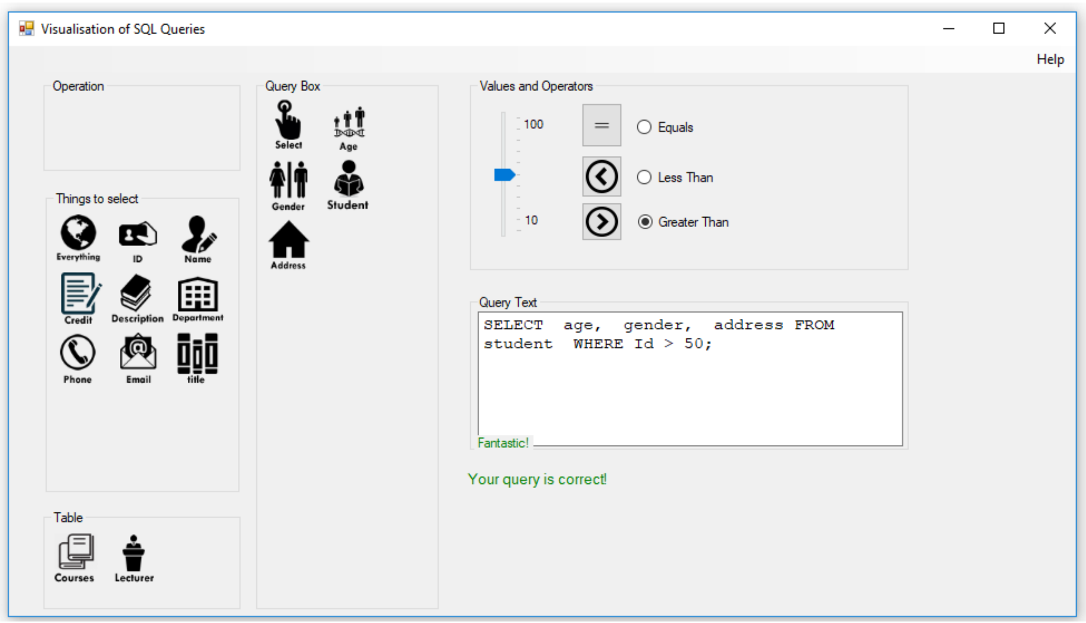


Figure 35: A successfully generated query

queries. This work will particularly aid undergraduate students who are learning SQL queries for the first time. Another application area of this work can be extended to commercial business systems, where the visualiser may be used to assist non-professional users comprehend SQL queries. While such users may be aware of databases, their knowledge of SQL queries may be limited. We believe that our technique's clear communication and visualisation-focus will help users to easily understand SQL queries.

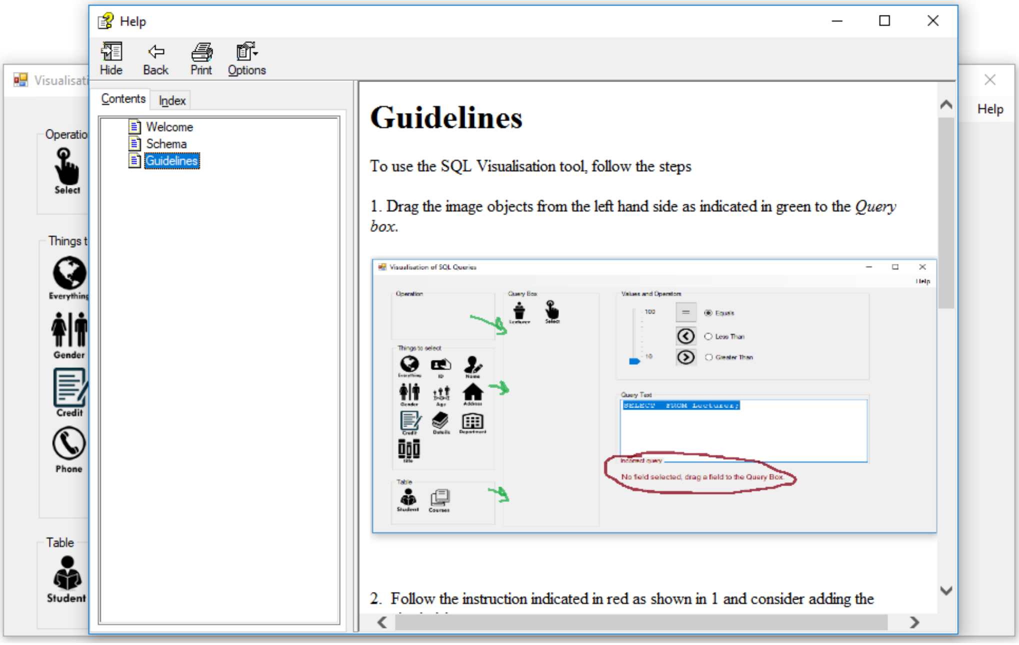


Figure 36: The help feature presents the guidelines for the user

7.4 CHAPTER SUMMARY

This chapter has presented an interactive visualisation tool that used visual specifications to build SQL queries. The visualisation tool considered the SQL `SELECT` constructs in a bid to improve the comprehension process. It is generally agreed that visualisation can encourage active participation and also lead to critical thought processes in students [Gray and Malins 2016; Lye and Koh 2014]. Hence, this work is consistent with those that have used visual specifications. In Chapter 8, we will examine another approach that converts speech into SQL queries.

8

SPEECH TO SQL QUERY

The preceding chapter presented the visual specification method that used the *drag and drop* interaction to generate a query. This method was implemented into a tool called the `SQL Visualiser` that uses images to generate a query. This chapter introduces the verbal specification technique into a speech-based query system named `TalkSQL` that takes speech inputs from a user, converts these words into SQL queries and returns a feedback to the user. Automatic feedback generation is of immense importance. To achieve this, we have used REs, a representation of regular languages for the recognition of SQL queries and automatically generate feedback using pre-defined templates.

8.1 INTRODUCTION

NLP has contributed immensely to the field of HCI, in terms of its theoretical results and practical applications. These applications have led to the emergence of robust speech-enabled user interfaces (VUIs) such as Google's Voice Action, Apple's Siri, Amazon's Alexa, Microsoft's Cortana etc [Feng *et al.* 2017; Zhang *et al.* 2018; Saha *et al.* 2019]. Together, these VUIs have been applied to solve real world problems in Healthcare [Shah and Sengupta 2018], Internet of Things (IoT) [Jungbluth *et al.* 2018], Military [Levulis *et al.* 2018], Telecommunication [Kapur *et al.* 2018], etc.

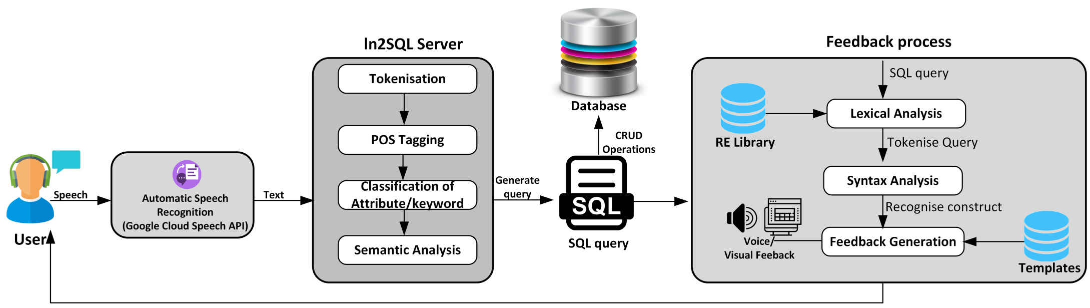


Figure 37: The framework of `TalkSQL`

This chapter introduces a speech-based, conversational NLIDB system called `TalkSQL`, used to aid the comprehension of SQL. `TalkSQL` is an intelligent, conversational tutoring system that translates natural language queries into an executable SQL query to be used on a test database, and presents an output. Similarly, a speech feedback is provided to a learner. `TalkSQL` applies the CRUD (`CREATE, SELECT(READ), UPDATE and DELETE`) operations, used against a relational database.

In Figure 37, we show the architecture of `TalkSQL`. A user initiates a verbal request to the tool, which is then processed and converted into text by the Google Cloud Speech API. The text version is preprocessed by the `ln2SQL` server that uses the `spaCy` NLP engine (described in Chapter 3) for conversion into a SQL query. The generated query is used on a relational database to request data. Automated feedback is an important aspect of this work. To provide a user with a feedback, the query is tokenised and grouped into syntactic parts. The recognised parts are then matched with a feedback template, and a feedback is generated. The feedback is presented to the user in textual, vocal and visual forms. The `TalkSQL` engine is similar to that presented in `ln2SQL` [Couderc and Ferrero 2015] which considers only the `SELECT` command.

8.2 FEEDBACK GENERATION

Feedback has shown to assist novices comprehend their programs and in many cases, applied in textual forms to provide insights [Ade-Ibijola *et al.* 2014; Singh *et al.* 2013]. As discussed in the literature (Chapter 3), different NLIDB systems have been able to provide feedback in different forms, but to our knowledge, none has undergone our approach. The technique we have developed uses REs to recognise SQL queries before generating a feedback to a user. The feedback is usually concise in a bid to assist learners understand SQL queries. Algorithm 14 shows spoken words from a user, the query equivalent is presented in Listing 14 and the expected feedback is displayed in Algorithm 15.

Algorithm 14 Spoken words from a user (NL)

Amend the student name to John whose id is equal to 6

Listing 14: SQL query to update a single record

```
UPDATE Student
SET name = 'John'
Where ID = 6;
```

Algorithm 15 Expected Feedback

```
You have updated a record with a name called John, whose ID number
is equal to 6 in the Student table.
```

The tool described in this study takes verbal inputs from a user, performs operations on it and generates a query. The feedback from the query generation is similar to that shown in Algorithm 15. The next section presents REs for the recognition of the CRUD commands syntaxes in SQL before generating a feedback to the user.

8.3 SQL CONSTRUCTS ABSTRACTION USING REGULAR EXPRESSIONS

In this section, we show how we have abstracted the CRUD SQL operations using regular expressions. This stage is regarded as the *Lexical Analysis* phase. This phase shows how the streams of characters (lexemes) that make up the language are grouped into tokens for recognition. To begin, we represent the CRUD operation with a diagram that describes how the building blocks of tokens should be formulated as seen in Figure 38. The CRUD commands

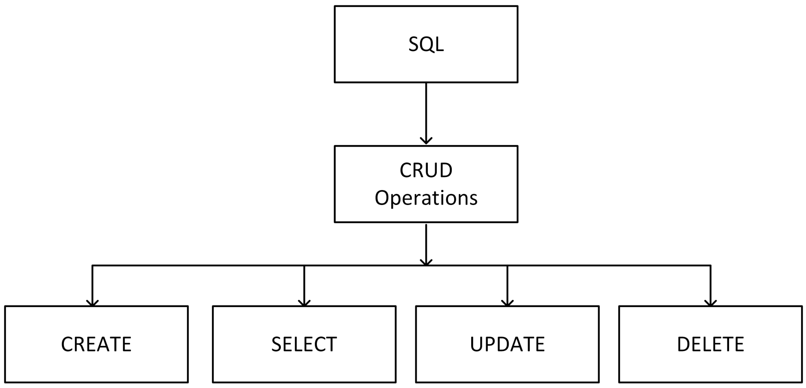


Figure 38: The diagram denoting the CRUD SQL commands

consists of four statements used to perform operations in a database. These statements are `Create`, `Select`, `Update` and `Delete`. These statements have been described using regular expressions in [Ade-Ibijola and Obaido 2017], as discussed in Section 4.3.

8.4 INTRODUCING TALKSQL

`TalkSQL` was designed as a C# Windows Forms Application (WPF) that runs on the .NET framework [MSDN 2017]. As a VUI, `TalkSQL` translates natural language specified in verbal inputs into an executable SQL query to be used on a test database and presents an output. Figure 40 shows the user interface of TalkSQL. A successful conversion will produce a result to a user in the form of a visualisation. An automatic feedback is also available to a user in textual and speech forms. The idea is to provide a comprehensive feedback to a learner. Errors are also handled by `TalkSQL` in the form of refinements. `TalkSQL` informs the user to refine their statement in a conversational manner and produces a result once the statement is understood. The schema for the test database is provided in Figure 39.

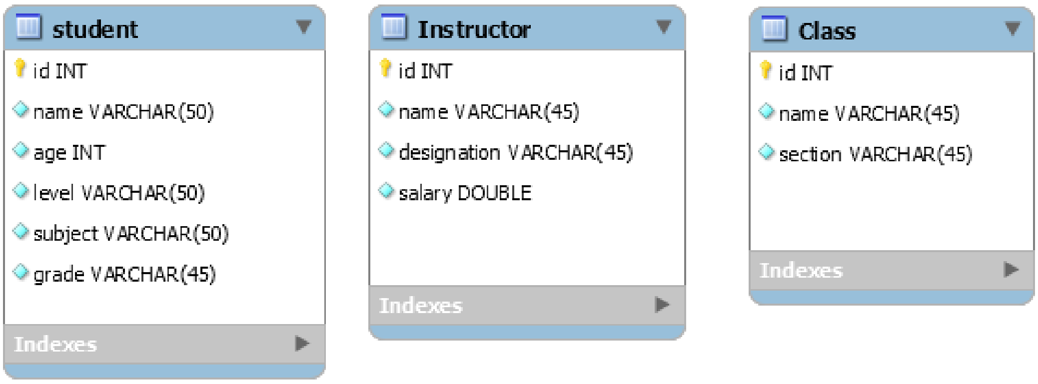


Figure 39: The schema of the test database

The schema contains three tables (school, instructor, class) with their associated attributes. Each table is linked by a primary key (*id*), which is the unique identifier. `TalkSQL` uses this schema to perform the query generation. It is worth-noting that `TalkSQL` can be adapted to any database. We have used this sample database to test the tool that has been developed for this work.

8.4.1 *Implementation and Results*

In this section, we show how `TalkSQL` (Figure 40) was implemented to translate speech inputs into a SQL query and the feedback is presented to the learner. Next, we describe the error-handling and how it can be refined to construct a query. Last, we present possible applications of `TalkSQL`.

8.4.2 *Query Translation*

In this section, we discuss the translation of user inputs from verbal inputs, before a query is generated and feedback is shown to a user. `TalkSQL` undergoes five phases before a query is generated as presented in Figure 22. In the first phase, words from a user are converted into text by the Google Cloud API service for further processing. This process is known as the speech recognition phase. The second phase takes the recognised text and match it to sequences of tokens. At this phase, whitespaces and noisy features such as comma, etc., are removed. This phase is known as the tokenisation phase. The third stage tags each tokenised sentence to their respective parts-of-speech (also known as POS Tagging). For example, the POS-tagging for the sentence will be:

VB	DT	NN	IN	DT	NN	NN
Display	*the*	*age*	*from*	*the*	*student*	*table*

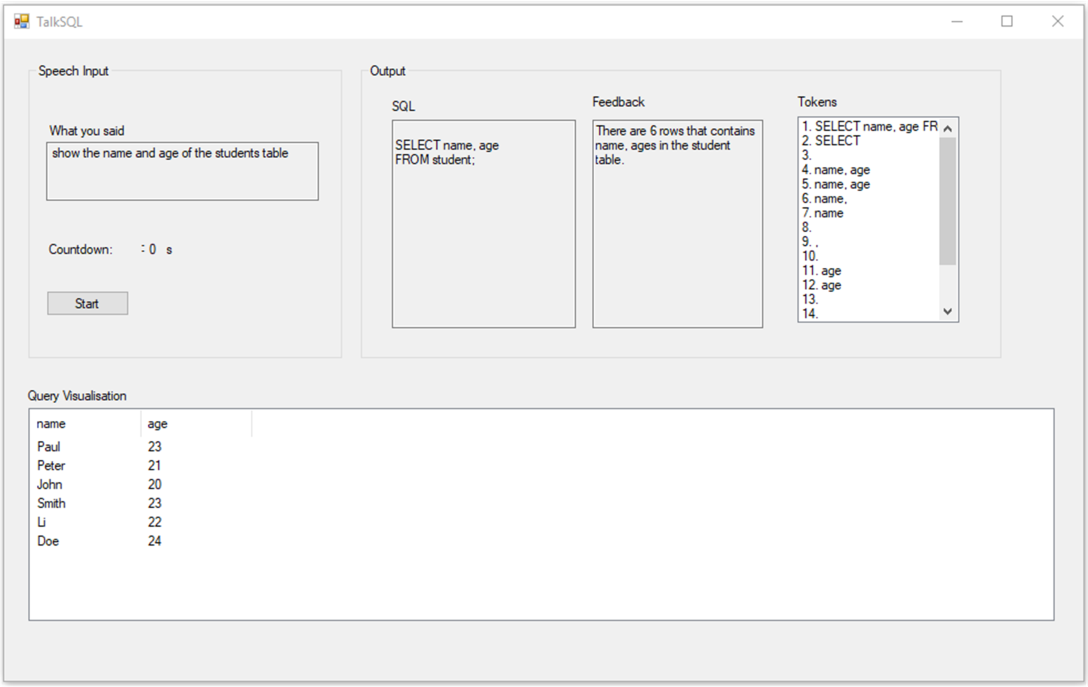


Figure 40: The User Interface of TalkSQL

In this example, NN stands for a singular noun, DT - determiner, IN - preposition and VB stands for a verb. The fourth phase classifies each word to their syntactic parts. This phase is regarded as the syntactic analysis phase where each word is transformed into structures and shows relations. The fifth phase analyses the structure from the syntactic phase and check for meanings before words are mapped to their equivalent meaning. For example, words such as select, choose, pick and display are mapped to SELECT keyword. Words such as create, produce, make, build are mapped to the CREATE keyword. Also words such as update and amend are mapped to the UPDATE keyword. For the DELETE keyword, words such as remove, delete are mapped. Once attributes and keywords are mapped, the query is generated and used in a database.

In this section, we have presented the query translation phase as described in Figure 22. The next section presents the result of the CRUD statement.

8.4.3 *Results of the CRUD Operations*

`TalkSQL` was successful in generating feedback for the CRUD statements. We start with the CREATE command as seen in Algorithm 16. This shows a request from a user. The converted SQL query is seen in Listing 15 and feedback provided in Algorithm 17.

Algorithm 16 Create command words from a user (NL)

Make a class table and specify ID as integer, name as alphanumeric entries with at most 45 characters and section as alphanumeric characters of at most 45 characters

Listing 15: CREATE SQL query

```
CREATE TABLE class (
ID int ,
name varchar(45),
section varchar(45)
```

```
);
```

Algorithm 17 Feedback for the CREATE statement

```
You have created a new table called class with the following
information, ID stores integer values, name stores alphanumeric
entries that contain 45 characters, section stores alphanumeric
entries that contain 45 characters
```

The natural language that matches a SELECT command is presented in Algorithm 18. The row count function, count(), is used to retrieve the number of rows in a table. We added this function to the feedback generator to display the number of rows that appear in a table. The converted query version is presented in Listing 16. The feedback for this query is presented in Algorithm 19.

Algorithm 18 Spoken words from a user (NL)

Show the name and age of the student table

Listing 16: SELECT SQL query

```sql
SELECT name, age
FROM student;
```

Algorithm 19 Feedback for the SELECT statement

```
There are 6 rows that contains name and ages in the student table
```

Algorithm 20 shows the update keyword specified in natural language spoken by the user to update a table. The NL query is converted into SQL queries in Listing 17. The corresponding feedback is presented in Algorithm 21.

Listing 17: UPDATE SQL query

```sql
UPDATE student
SET name = 'John'
Where ID = 6;
```

Algorithm 22 shows the delete keyword specified in NL, spoken by a user to delete a record from a table. Listing 18 shows the converted SQL query. The feedback is presented in Algorithm 23.

Listing 18: DELETE SQL query

```sql
DELETE FROM lecturer
Where name = 'John';
```

8.4.4 *Error-handling and Refinement*

To handle errors, `TalkSQL` uses two phases to resolve any ambiguity. The first phase requests a user to supply a missing attribute and table information, while the second phase finds missing keywords from a user statement.

8.4.4.1 *Table and Attribute Ambiguity*

`TalkSQL` handles errors that may occur in a conversational manner. If some information is required before a query can be generated, `TalkSQL` notifies the user to complete the statement

Algorithm 20 Spoken words from a user (NL)

Amend the student name to John whose id is equal to 6

Algorithm 21 Feedback for the Update statement

```
You have updated a record with a name called John, whose ID number
is equal to 6 in the Student table.
```

with the missing information using a speech-based conversation. For example, if the user says: "Show me the names in a table". It can be seen that the *name* attribute appears in the three tables as specified in the schema in Figure 39. `TalkSQL` uses this template to notify the user:

Do you mean the {attribute} in the `Table 1 | Table 2 | ...| Table N` table ?

In this case, the `TalkSQL` would ask the user: "Do you mean the name in Lecturer, Student or Class table?". Once the user clarifies the missing table. `TalkSQL` proceeds with generating the SQL query and the feedback is provided to the user. Similarly, if a user makes a request such as "Display all details from the table" for a query to be generated and does not supply a table name, we use the template to request a user to supply the required table:

Do you mean the `Table 1 | Table 2 | ...| Table N` table ?

As specified in the template, the request would be: "Do you mean the Lecturer, Student or Class table ?". Here, `TalkSQL` would ask the user to complete the statement and once provided, a query will be generated.

8.4.4.2 *Keywords*

In situations where the keywords such as table and attributes information are missing, `TalkSQL` requests a user to provide additional information. For example, if a user requests: "Show the information from the table". `TalkSQL` uses this template to request the user to provide additional details before generating a query:

Could you specify which {Table}, and its {Attributes} ?

Once the information required is provided by the user, `TalkSQL` generates the query. We compared `TalkSQL` with a number of NLIDB systems as presented in Table 11.

8.4.5 *Applications of TalkSQL*

In this section, we present possible applications of `TalkSQL` that were introduced in Section 8.4. `TalkSQL` may find applications in:

1. QA systems: As a tool, it may be used to provide a solution to a user's question specified in natural language.
2. learning aids: It may be applied to assist users understand and improve their cognition of the SQL concepts.
3. ITSs: `TalkSQL` may be used to provide immediate feedback to a learner without relying solely on an instructor. In addition, it can be used as a practice aid to assist learners understand SQL.
4. assistive technologies: For a visually impaired learner, `TalkSQL` may be used to enhance SQL learning, since it is hands-free and provides feedback in speech forms.

Algorithm 22 Spoken words from a user (NL)

Remove a record from the lecturer table where the name is John

Algorithm 23 Feedback for the Update statement

```
You have deleted 1 record from the lecturer table where the name is John
```

Table 11: Existing tools versus `TalkSQL`

	Focus	Purpose	SQL Commands				Feedback		
	Target (Novice)	Teaching and Learning aid	CREATE	SELECT	UPDATE	DELETE	Speech	Text	Visuals
LUNAR	✓	✓	✗	✓	✗	✗	✗	✗	✓
MaNaLa	✓	✓	✗	✓	✗	✗	✗	✗	✓
EchoQuery	✓	✓	✗	✓	✗	✗	✓	✗	✓
SpeakQL	✓	✓	✗	✓	✗	✗	✓	✗	✓
Cyrus	✓	✓	✗	✓	✗	✗	✓	✗	✓
Ln2QL	✓	✓	✗	✓	✗	✗	✗	✗	✓
TalkSQL (Our tool)	✓	✓	✓	✓	✓	✓	✓	✓	✓

5. improving SQL comprehension: Generally, as a tool, it may be used by non-technical end-users in industry and students in higher institutions of learning to improve their SQL understanding.

8.5 SCOPE AND LIMITATIONS

In this section, we present the scope and limitations of `TalkSQL`. Up until this point, `TalkSQL` can perform error-checking and provide feedback for simple CRUD operations. The tool uses REs for the recognition of the simple CRUD operations in SQL. A noticeable feature that `TalkSQL` cannot handle is generating feedback for queries in complex forms – especially queries enclosed with balanced parentheses (See Listing 19).

Listing 19: A non-regular nested SQL query statement

```
SELECT  firstname
FROM    employee
WHERE   empid IN (
                SELECT DISTINCT
                empid
                FROM
                employee);
```

The example in Listing 19 contains two nested SELECT statements. The inner query within the parentheses displays only the empid information. The outer query displays the firstname from the employee table. These types of queries can only be recognised by a CFG – a type of irregular language. It is interesting to note that REs are only used for lexical analysis tasks rather than syntactic parsing. A parser generator such as Coco/R [Mössenböck 2005] will fix this issue.

8.6 CHAPTER SUMMARY

In this chapter, a speech-based tool called `TalkSQL` was introduced, that takes speech inputs from a user, converts these words into SQL queries, and then returns a feedback to a user. We improved an existing NLIDB framework to accommodate the CRUD operations which allows a user to create, retrieve, modify and delete data from a database. `TalkSQL` uses REs to recognise these SQL operations and generate a feedback to be shown to a user. `TalkSQL` was able to recognise simple query commands that do not contain balanced parentheses. As a tool, it may find applications in real-world scenarios and improve SQL learning.

Part iv contains two chapters. The first chapter, Chapter 9, contains the evaluation results of the prototypes developed from Chapter 4 to Chapter 8. The conclusion of this thesis is presented in Chapter 10.

Part IV

EVALUATIONS, CONCLUSIONS AND FUTURE WORK

Throughout this study, we have presented a number of algorithms together with some software prototypes that assist end-users comprehend SQL queries. These prototypes are (i) `S-NAR`, a narrator for explaining SQL queries using predefined templates; (ii) `SQL-Narrator`, an improved SQL narrator for nested queries; (iii) `Narrations-2-SQL`, a tool that translates a narration into SQL queries; (iv) `SQL Visualiser`, a visualisation tool that uses the drag and drop interaction for query generation; and (v) `TalkSQL`, a speech-based query tool, that converts speech into SQL queries. For each prototype, we examine end-users' perceptions and form a basis for future studies. Many evaluations were conducted using an online means, and where necessary, we undertook some performance evaluation.

This part consists of two chapters. In Chapter 9, we present the evaluation results for the prototypes designed in this study and provide the conclusion with discussions for future work in Chapter 10.

9 EVALUATION OF PROTOTYPES

9.1 INTRODUCTION

Throughout this study, we have developed a number of prototypes from Chapter 4 to Chapter 8. These tools have contributed immensely to the problem of SQL understanding and synthesis. In this chapter, we present the result of the evaluation that was conducted for each of the studies. Majority of the evaluation was conducted through an online survey designed as questionnaires. These questionnaires were designed using Google Forms. A link was sent to all participants at the university for their contribution. The questions used for these surveys are provided in Appendix A.

9.2 EVALUATION OF S-NAR

This section presents the accuracy of the `S-NAR` tool. `S-NAR` was tested on 5000 queries scrapped from the Internet (indexed in Appendix E), and it successfully narrated a subset of these queries (96%). This is presented in Equation 38.

$$\text{Accuracy} = \frac{4824}{5000} = 96.48\% \tag{38}$$

We noticed that the remaining 4% of the queries were nested queries and had balanced parentheses in them. This could only be recognisable by a CFG or higher classes of formal abstract machines. This was already addressed in Chapter 5.

9.3 EVALUATION OF THE SQL NARRATOR

In this section, we present the result of the evaluation carried out using an online survey from 161 students at the University of the Witwatersrand. The participants were mostly undergraduate CS students at the University of the Witwatersrand whom had already been taught simple and nested queries. Participation was strictly non-mandatory and participants' profiles were kept anonymous. The survey is available in the link: `https://goo.gl/CQcVZN`.

9.3.1 *Result of the Survey*

Out of the 161 responses received, 97.5% claimed they are familiar with SQL queries, 1.2% indicated no familiarity and 1.2% were unsure about their response - this is presented in Figure 41. In addition, we asked the participants if they think nested queries are difficult and 96.3% agreed that nested queries are difficult, 8.5% claimed they do not think nested queries are difficult, while 3.4% were unsure about their responses (see Figure 42).

A total of 98.1% agreed that they were able to comprehend nested queries using the `SQL Narrator` and 1.9% claimed that they could not comprehend the nested queries using the narrator (in Figure 43). Furthermore, we asked the participants to answer which of the generated narrations they were able to comprehend (see Figure 44). About 91.4% strongly agreed that the outer to inner subquery narration was easier to comprehend while 1.9% chose the cojoined subquery narration, and 6.8% agreed with the inner to outer subquery narration. It is interesting to note that the majority of the participants agreed with the outer to inner narration for the subquery. Perhaps, they were comfortable with the chronological flow of the explanation of the subquery. In addition, we asked the participants to rate the `SQL Narrator` on a scale of 1- 10; it was seen that majority agreed that the tool was useful to them (see Figure 45).

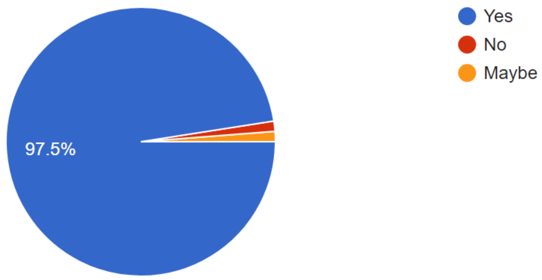


Figure 41: Knowledge of simple queries

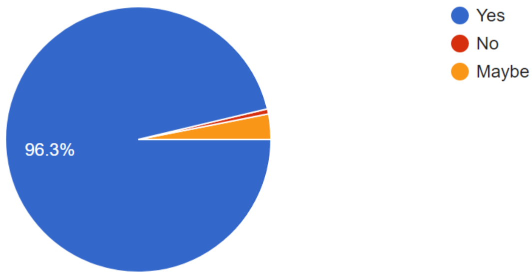


Figure 42: Are nested queries difficult?

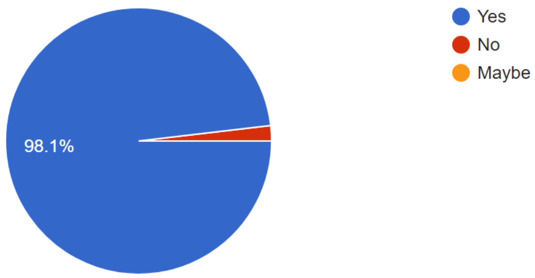


Figure 43: Comprehend nested queries using the SQL Narrator

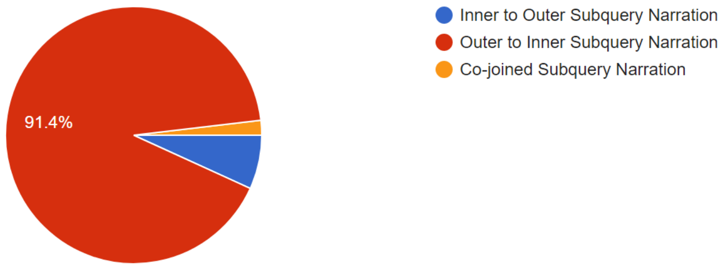


Figure 44: Comprehend which of the generated narrations

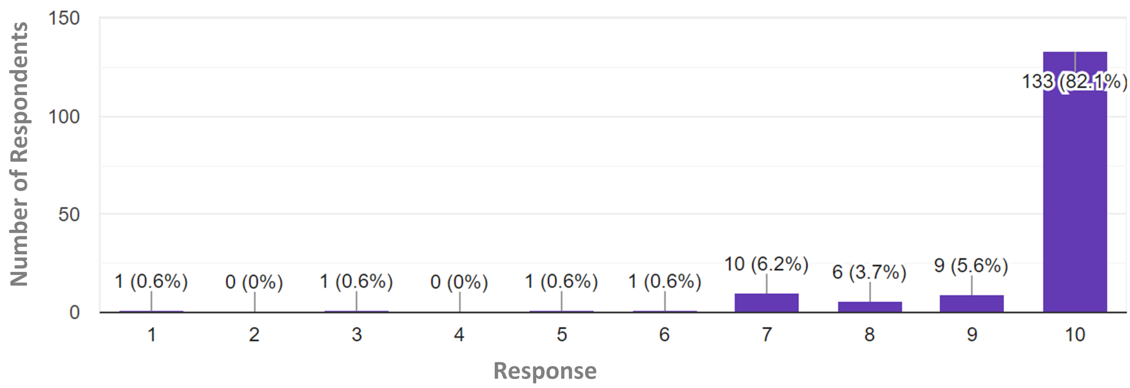


Figure 45: Rate the SQL Narrator

9.3.2 *Respondent Feedback*

The last part of the questionnaire was open-ended. We asked the respondent to provide any recommendation they might have for us to improve the tool. Some of the recommendations provided by the participants are given in this section. With the result of the evaluation, and the recommendation provided, we can conclude that adopting this tool will improve student comprehension of nested queries. Some comments that were anonymously given by the participants are as follows.

> "I would not suggest anything, as the tool makes nested queries very easy to understand."

> "This tool is easy to understand compared to other narration tools that I have used."

> *"It's much nicer to see what the queries are, and explanations of each of them were provided."*

> *"It was easier to read the sentences and follow it through. I am sure the narrator will be more useful with more complicated examples, this will benefit a novice programmer altogether."*

> *"Nothing to suggest. It seems to do exactly what it's meant to and does it really well. I liked the fact that the narrations are well structured sentences and aren't complicated explanations."*

9.4 EVALUATION OF NARRATIONS-2-SQL

The evaluation was carried out in a two-fold manner: (1) Using the crowdsourced XNorthwind dataset indexed in Appendix D, we show the accuracy of the `Narrations-2-SQL` tool. (2) Using human subjects, we show the users' perceptions of the tool we have designed for this study. The survey can be accessed through `https://bit.ly/2m62guw`.

9.4.1 *Accuracy of Narrations-2-SQL*

We used the crowdsourced XNorthwind dataset to train our tool. The end-users were asked to test `Narrations-2-SQL` with their narrations. We discovered that about 180 narrations from the end-users were able to successfully match an SQL query. To determine the accuracy, we take:

$$\text{Accuracy} = \frac{180}{204} = 88\% \tag{39}$$

It is worth noting that the decrease in the accuracy was due to some of the narrations being outside are out of our JFA training data. In future work, we will improve our JFA to recognise more queries by adding more keywords and semantic rules.

9.4.2 *Survey Design*

The survey was carried out through an online means and feedback was received from 162 participants. The results of the survey in this section and was strictly anonymous. The survey can be accessed via `https://shorturl.at/aJMN8`.

9.4.2.1 *Result of the Survey*

A total of 162 responses were received, and 88.9% agreed that they were familiar with SQL, 8.0% admitted no familiarity of SQL and 3.1% were unsure (see Figure 46). In addition, the participants were asked if they thought `Narrations-2-SQL` is user-friendly; 98.8% agreed that the tool was user-friendly and 1.2% were unsure about their responses (see Figure 47). They were also asked if they thought the generated SQL query was a correct translation of their narrations; about 93.8% admitted yes and 6.2% were unsure about their responses (see Figure 48).

When the users were asked if the tool will help end-users in industry with no knowledge of SQL, 96.9% of them agreed that the tool will be helpful to industry users and 3.1% were unsure (in Figure 49). In addition, we asked the participants to rate the tool that we have developed on a scale of 1-10. The result is presented in Figure 50.

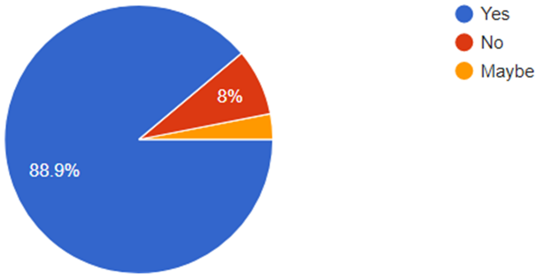


Figure 46: Familiar with SQL

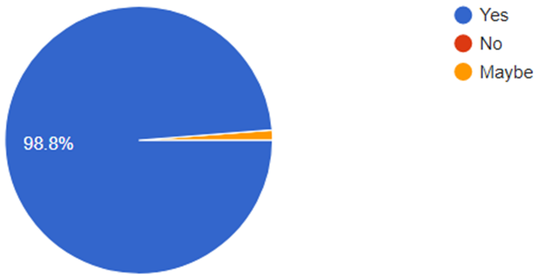


Figure 47: The user friendliness of Narrations-2-SQL

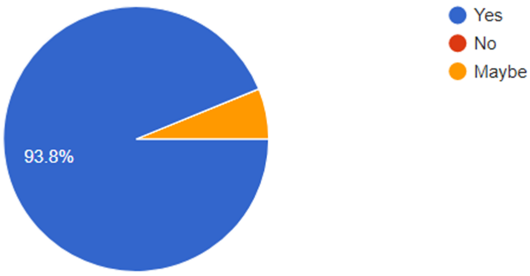


Figure 48: The generated SQL query a correct translation of narrations

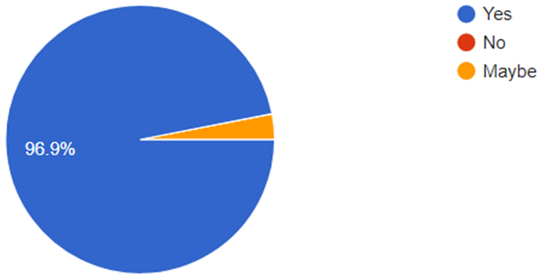


Figure 49: The tool will help end-users in industry with no knowledge of SQL

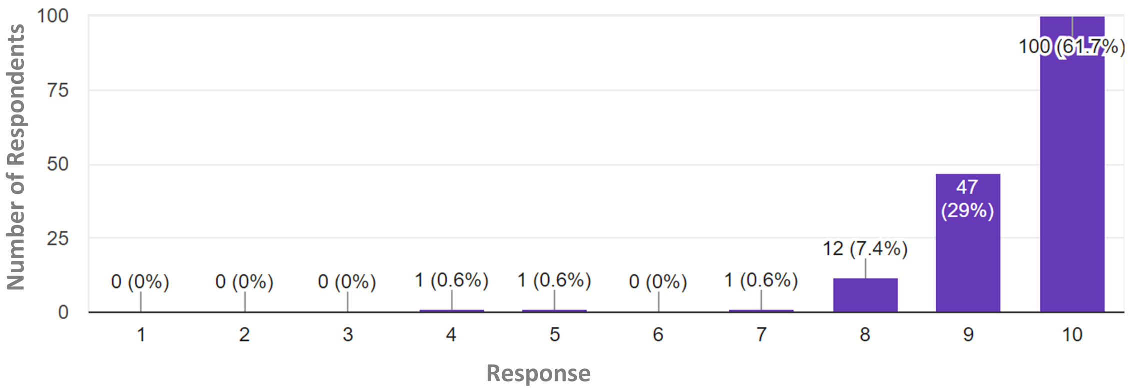


Figure 50: Rate the Narrations-2-SQL

9.4.2.2 *Respondent Feedback*

In this section, we capture participants' comments about the `Narrations-2-SQL`. Open-ended questions were asked to get participants to get their insights about what they thought of `Narrations-2-SQL`. It was worth noting that the feedback provided was interesting and would enable us improve `Narrations-2-SQL` for future research. Some comments from the participants are as follows.

> *"This is a very helpful tool and will improve a lot of successful my SQL code creation skills."*

> *"Great to use and very simple. It will be easy for people that do not even know much about SQL"*

"I think this tool will help eliminate a lot of errors in SQL coding"

"I would honestly recommend this tool to anyone who intends to queries in SQL"

"I have two observations: Could you include voice prompt so that it will be easier to use, and could you increase the font size of the query result."

With these results, we conclude that automating narrations into SQL will be beneficial for end-users.

9.5 EVALUATION OF SQL VISUALISER

The evaluation of the `SQL Visualiser` was carried out using an online survey from 121 students from the University of the Witwatersrand. The respondents were mostly undergraduate CS students and majority of them had knowledge of SQL. The questionnaire is split into two parts: the first required the students to answer general questions about their knowledge of visualisers, while the second focused on their perception of the `SQL Visualiser` we have designed in this research. In addition, the students were asked to provide feedback on ways to improve the `SQL Visualiser`. Constructive feedback was received. The survey is available in the link: `https://bit.ly/2m1Pf4Y`.

9.5.1 *Result of the Survey*

Out of the 121 responses, 89.3% admitted to have knowledge of SQL, 7.4% affirmed no knowledge of SQL and 3.3% were unsure about their responses – this is presented in Figure 51. Of the participants, 94.2% agreed that the `SQL Visualiser` was user-friendly, 4.1% admitted that the visualiser was not user-friendly, and 1.7% were unsure about their responses (see Figure 52).

Furthermore, the students were asked if they were able to synthesise basic SQL queries using the visualiser (in Figure 53). About 95% agreed that the visualiser helped them comprehend SQL queries, 4% admitted that they find the visualiser difficult to use and 1% stayed indifferent. In addition, 92.6% admitted that visual specifications helped them understand the syntax of the SQL queries, 5% did not agree and 2.4% stayed indifferent (see Figure 54). We asked the participants to rate the `SQL Visualiser`; their responses are captured in Figure 55.

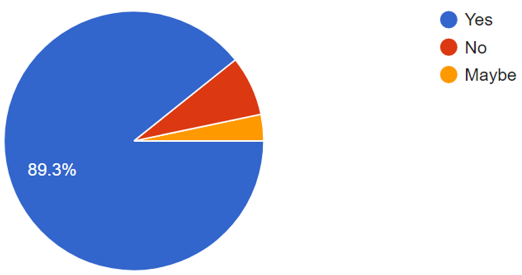


Figure 51: Knowledge of SQL

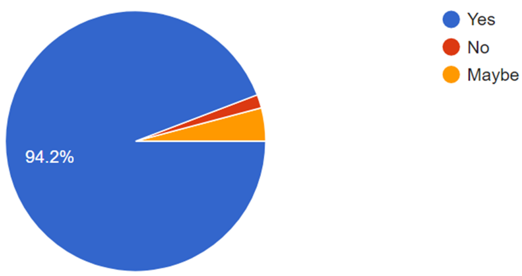


Figure 52: The user friendliness of the SQL Visualiser

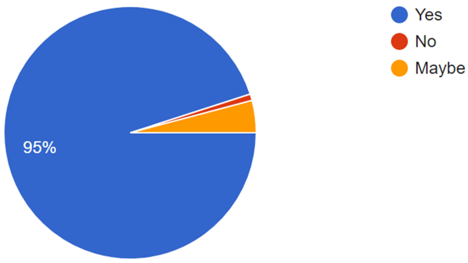


Figure 53: Ability to synthesise basic SQL queries using the visualiser

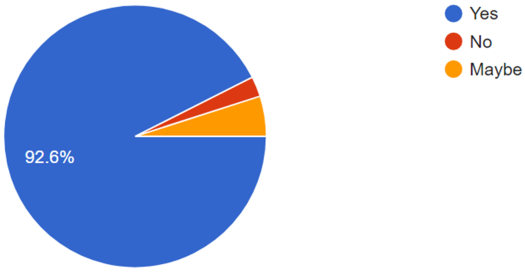


Figure 54: Visual specifications helped comprehend SQL

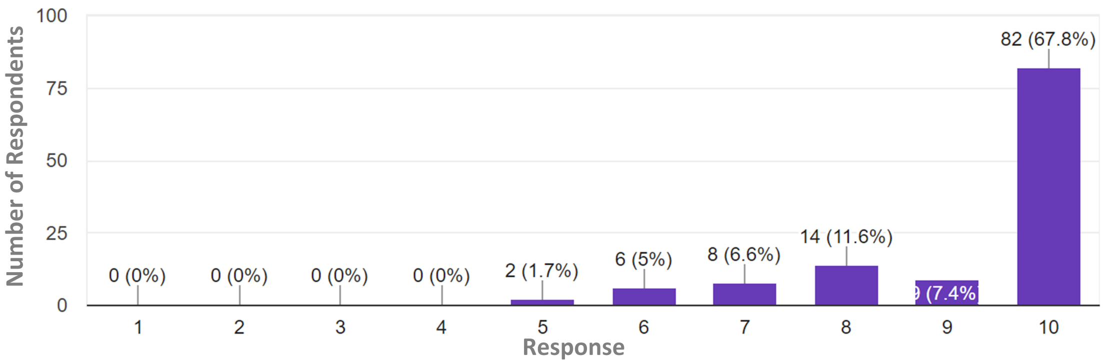


Figure 55: Rate the SQL Visualiser

9.5.2 *Respondent Feedback*

The feedback the respondents made was helpful and some respondents highlighted some limitations of the `SQL Visualiser`. Examples of the positive comments made were:

> *"The visualiser was extremely helpful. I liked the way the pictures assisted in displaying the query. As a learner, I think it will help other students understand SQL queries better and improve our knowledge of the SQL concept."*

> *"Icons were a good idea and helpful. It is not boring."*

> *"Far much better visualiser l have used so far."*

> *"It seems simple and straight-forward, the user does not have to try hard to understand its functionality and operation."*

Some of the limitations mentioned by the respondents include:

> *"There should be explanations, the use of comments in the query text box would be extremely useful. I have used SQL before, therefore this is mostly targeted at novice users. Other methods should be integrated to cater for expert users."*

> *"The icons colour choice is boring."*

"Possibly increase text sizing for the visually impaired user."

"This tool should allow users to choose their own icons (e.g. students could be a different icon)."

The majority of the students commended the use of visual specifications to aid their comprehension. These results are consistent with the evaluation carried out on a visualiser [Satyanarayan and Heer 2014] for program comprehension, where users perceptions supported the usefulness and importance of visual specifications. We believe that adopting this tool in higher institutions of learning will improve students' comprehension of SQL.

9.6 EVALUATION OF TALKSQL

The survey was carried out online and the feedback was received from 113 participants. The majority of the participants were undergraduate CS students taking a database course and most of them were familiar with SQL. The survey can be accessed through `https://bit.ly/2ktYNVX`. The result of the survey is presented in the next section.

9.6.1 *Result of the Survey*

We received a total of 113 responses, out of which 98.2% agreed that they had knowledge of SQL, 0.9% were unsure and 0.9% indicated no knowledge of SQL – this is shown in Figure 56. Furthermore, we asked the participants if the `TalkSQL` tool was easy to use; 98.7% claimed that the tool was easy to use and 5.3% were not sure of their responses (see Figure 57). About 91.2% claimed that they were able to understand the CRUD commands, 8% were indifferent and 0.9% claimed they could not understand the CRUD operation using `TalkSQL` (see Figure 58). Next, we asked the participants if they think a visually impaired learner could understand SQL with `TalkSQL`, a total of 87.6% agreed, 10.6% were unsure and 1.8% do not agree that this category of learners would understand SQL query using `TalkSQL` (in Figure 59). We asked the participants to rate TalkSQL, their responses are indicated in Figure 60.

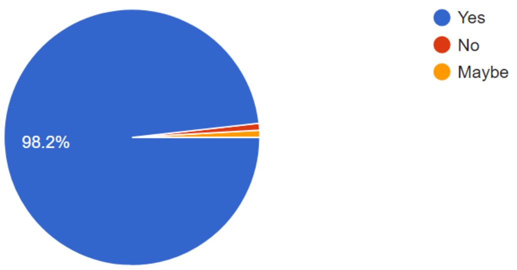


Figure 56: Knowledge of SQL

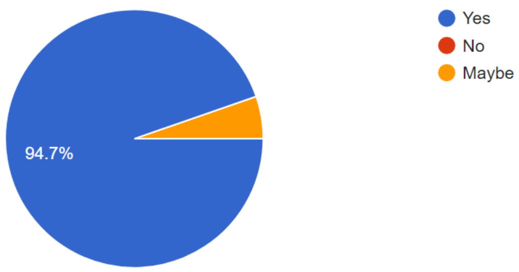


Figure 57: Ease of use of TalkSQL

9.6.1.1 *Respondent Feedback*

Open-ended questions were asked from the participants to indicate what we could do to improve `TalkSQL`. The feedback received was quite helpful, while some highlighted a couple of limitations. Some positive feedback received are as follows:

"It is really nice and very helpful. It will help learners who are colour-blinded to learn SQL."

"I liked the part where a voice feedback was read back to me, no need to type!."

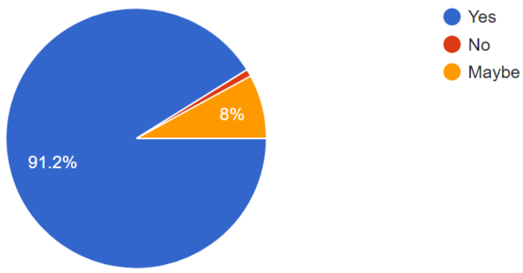


Figure 58: Able to understand CRUD operations using TalkSQL

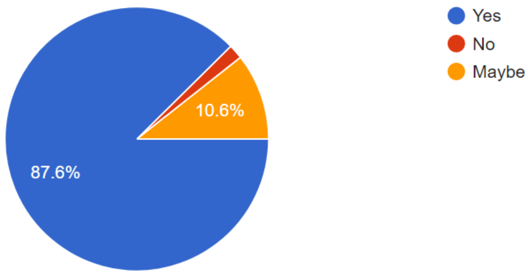


Figure 59: Able to assist the visually impaired understand SQL

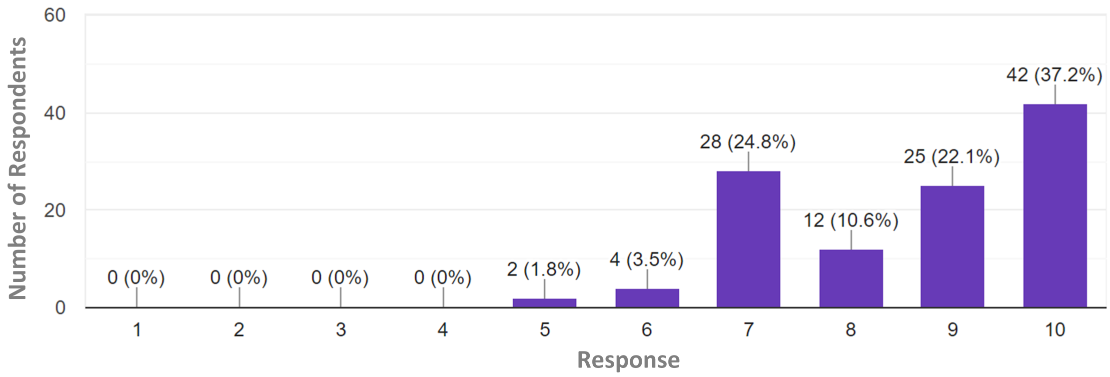


Figure 60: Rate the TalkSQL tool

> *"Very interesting tool! I could rephrase my words over again. Nice!."*

> *"No suggestions. TalkSQL works perfectly."*

Some limitations mentioned were:

> *"Please, increase the font size!"*

> *"What if my voice is strained, I think you should add typing features."*

> *"This tool should cater for nested SQL queries as well."*

Indeed, the feedback received will help us improve `TalkSQL` and make it accessible for learners that require its services. These results are consistent with the survey by Wilson [Wilson *et al.* 2010] where the participants agreed that verbal specifications can assist learners understand SQL. We are very positive that adopting this tool will stimulate students' interest of SQL in higher institutions of learning.

9.7 CHAPTER SUMMARY

This chapter presented the evaluation of the prototypes that were designed in this research. In the first prototype, we measured the accuracy and presented some results. Next, we conducted online surveys for the remainder of the prototypes, as seen from Section 9.3 to

Section 9.6. Overall, the feedback received on each survey indicated that the participants agreed that the tools stimulated their interests in the SQL concepts.

Chapter 10 concludes this research and provides ideas for future research.

10 CONCLUSION AND FUTURE WORK

10.1 CONCLUSION

Learning and writing correct SQL queries have shown to be significant problems in the academic environment as well as in industry. These problems have a seemingly unending impact on students and non-technical end-users alike. Even when there are highly skilled instructors, who are experts in the field, their solutions may not address the issue adequately. In most universities, a class may be large and an instructor might be less responsive to every student's needs. Similarly, this may occur in the industry whereby a developer who is saddled with all the technical intricacies of a company may leave the organisation, leaving the less-skilled user to scramble for solutions to their query needs. Consequently, such users might opt for online forums as a last resort, which may offer little or no help to their immediate query needs. Therefore, the overall problem is two-fold:

1. A student may be willing to learn SQL, which is paramount to passing a course and it is of equal importance when it comes to employability, but lacks sufficient knowledge to do so.
2. A non-technical end-user may have an urgent need to write queries as part of a routine task, but lacks the skill.

This thesis presented *SQL Comprehension and Synthesis* to address the aforementioned challenges. Additionally, it suggests interactive learning aids in an attempt to assist these choices of users to enable them to understand and write correct queries. Since the ultimate goal of learning aids is to enhance the teaching and learning process, this research is aimed at the improvement, and the comprehension process utilised by interactive aids. Ultimately, as stated in this research, interactive aids aim to improve the understanding of SQL queries, an area that students and non-technical end-users often struggle to comprehend. If implemented on a large-scale, the tools may be applied in numerous applications in real world scenarios and to improve SQL learning. The following prototypes have been presented in this thesis:

S-NAR. Chapter 4 introduced `S-NAR`, a tool that used REs in its engine to recognise SQL queries for the purpose of improving SQL comprehension. The tool translated the recognised SQL queries into textual explanations called narrations. This information is presented to a learner in real-time. In addition, the tool could serve as a learning aid by students or non-technical end-users that require explanations to queries written by technical staff. Similarly, the tool could be used to support teaching in line with the SFA to programming language pedagogy [Fincher 1999; Ade-Ibijola *et al.* 2014]. Another major advantage seen through the use of `S-NAR` is that it provides immediate feedback about the correctness of a query. `S-NAR` was tested with 5000 queries scrapped from the Internet, and it successfully reported an accuracy of 96%, which was a major success. In its current formation, the tool is unable to recognise nested queries enclosed in balanced parentheses. However, this may only be addressed by using an irregular language such as CFGs or higher classes of formal abstract machines, as presented in Chapter 5. `S-NAR` has appeared in Ade-Ibijola and Obaido [2017].

SQL NARRATOR. In Chapter 5, a CFG was designed for the automatic generation of narrations for nested SQL queries. This was implemented using Coco/R, a compiler generator that takes an attributed grammar and generates a scanner and a parser. These generated elements, both the scanner and the parser, were used to verify the correctness of a nested query. The designed grammar was implemented into a tool called `SQL`

Narrator based on the C# language. This further runs on the .Net framework. SQL Narrator was tested with the remaining 4% of queries that could not recognised by S-NAR in the previous study in Chapter 4. The tool successfully translated these queries. This idea has appeared in Obaido *et al.* [2019a]. The idea of using a CFG in this study was extended to the synthesis of 100000 hypothetical datasets that are similar to the Northwind database. The resulting database was referred to as XNorthwind (Extended Northwind) and has appeared in an extended study in Ade-Ibijola and Obaido [2019]. This database was used to train the tool, discussed in Chapter 6.

NARRATIONS-2-SQL. Chapter 6 proposed an approach that uses a JFA – a type of Finite Machine for translating natural language descriptions into SQL queries, which then further executes the queries, as well as provides feedback to a user. This technique was implemented into a tool called Narrations-2-SQL. This idea has appeared in Obaido *et al.* [2019b] An experimental evaluation was performed on 204 crowdsourced queries in natural language from the XNorthwind DB. The result thereof reported an accuracy of 88%. This report revealed that there is room for improvement. To our knowledge, this is expected to be the first time in which such an approach would be applied for SQL query translation from natural language. Since a natural language is context-sensitive, the JFA approach has shown to be an effective technique to the problem of query translation from natural language. If implemented on a large scale, Narrations-2-SQL may assist end-users in different domains to specify queries in natural language and perform tasks seamlessly without requiring much help from technical users.

SQL VISUALISER. Chapter 7 presented an approach which made use of images that depict SQL commands to generate a query. This was designed into a tool called the SQL Visualiser. The visualisation technique ensured the interaction between visual specifications to build queries. This is expected to eliminate the need to memorise database schemas, which is a major problem faced by students while learning SQL. SQL Visualiser used visual specifications for 'drag and drop' interactions for generating SQL queries. So far, the tool is only able to generate queries within the SELECT command. An extended visualiser is anticipated to recognise more commands to support the JOIN, ORDER BY, GROUP BY and aggregate functions. SQL Visualiser has appeared in Obaido *et al.* [2018].

TALKSQL. In Chapter 8, a speech-based query system called TalkSQL was designed to assist end-users to specify queries using speech inputs. TalkSQL has appeared in Obaido *et al.* [2019c]. This tool relied on an existing framework which makes use of the spaCy NLP engine to recognise SQL commands. For speech translation, TalkSQL uses the Google Speech API that incorporates a Deep Neural Network, alongside the HMM to transcribe speech. This speech engine was chosen due to its WER of 9% which outperformed other automatic speech recognition engines as discussed in Section 3.7.3.4. For query explanation, TalkSQL uses the narration technique that automatically generates using regular expressions as described in Chapter 4. Currently, the tool is unable to provide explanations for queries in nested forms. We anticipate a narrator engine developed using Coco/R in Chapter 5, which could be used to fix this hitch.

Finally, an evaluation of the prototypes was provided in Chapter 9. During the first study, only an experimental evaluation was performed to determine the accuracy of the tool. This showed an accuracy of 96%. The second study reported that out of 161 participants, 98.1% agreed that the tool enabled them understand nested queries. In the third study, an accuracy of 88% was reported as experimental evaluation and 96.9% out of 162 participants agreed that the tool would be helpful to industry users. The fourth study showed that out of 121 responses, 92.16% indicated that the tool aided their understanding of the SQL syntax. In the fifth study, out of 113 responses, 87.6% acknowledged that the tool would certainly help visually impaired learners to correctly write queries using voice inputs. The next section presents discussions for future directions.

10.2 FUTURE WORK

The prototypes developed in this thesis provide a basis for future research in several areas. At least five such areas can be identified. These areas include:

1. Extending `narrations` to other query forms.
2. Improving the `SQL Visualiser` to accommodate other SELECT statements such as JOIN, ORDER BY and other aggregate functions.
3. Extending the `Narrations-2-SQL` tool to work with multiple different databases.
4. Developing a practical quiz engine for SQL quiz grading.

The following sections elaborate the areas that have been highlighted in more detail.

10.2.1 *Extending Narrations*

This work has used narrations for aiding the comprehension of SQL queries. An area of exploration might be to abstract these explanations into a form that is free from language keywords. For example, we have used 'alphanumeric' to represent datatypes. It may be worth noting that some end-users may not clearly understand what the term 'alphanumeric' means. One way of capturing such a detail might be to use terms such as: 'letters and numbers'. These terms are well-defined in an end-user's vocabulary. Another interesting area of application of narrations to might be to explain queries that appear in other forms such as XML, SPARQL, JSON, etc. These will promote a clearer understanding of these queries.

10.2.2 *Improving the Visualiser*

The developed tool was able to easily generate simple queries using images. The next plan of action is to extend the visualiser to support other `SELECT` operation statements such as `JOIN, ORDER BY, GROUP BY` and aggregate functions. It will be interesting to see how learners could use these images to generate queries that contain these kinds of statements. Another desirable application of the visualiser is to generate nested SQL queries. These types of queries have shown to be problematic for students. Such solutions will help users to learn nested queries using such a visualiser. Other areas of exploration are to allow students to define their schema. This will provide a richer learning experience for the more advanced learner. It will be interesting to see how the extended visualiser can be made to generate queries to manipulate data in a database, and then produce a result. This will promote a more 'realistic' use of the tool. We have seen that the `Narrations-2-SQL` and `TalkSQL` use this approach on a live database and produce an output, which students find very useful.

10.2.3 *Extending the Narration-2-SQL Tool*

We have used the `Narrations-2-SQL` tool to work with the XNorthwind DB, which shows a good accuracy. It will be worthwhile to investigate the tool to work with multiple different databases such as Geo880, Academic, British National Corpus and IMDB to determine its accuracy. This may be an interesting problem to investigate. Recent studies by Baik *et al.* [2019]; Cai *et al.* [2017]; Yaghmazadeh *et al.* [2017a] have tested their approaches on multiple different databases. It will be interesting to compare our results with these studies. Additionally, it will be beneficial to use the JFA technique to other domains that require natural language specifications to generate code. For example, JFA can be applied to map query stored in JSON, XML, SPARQL and NoSQL formats.

10.2.4 *Developing a Quiz Game*

REs are useful for pattern matching and have been demonstrated in this research. It will be worth investigating this approach for a game-based scenario for learning SQL. Such an application may be useful to test learners' query skills.

Part V

APPENDIX

This appendix contains materials that are supplementary to contents that have been discussed in this thesis. This part has been organised as follows: the questionnaires used for the survey design are presented in Appendix A, Appendix B contains the REs library written in the Microsoft .NET framework, the atg file used for the grammar design by the Coco/R parser generator is presented in Appendix C, the crowdsourced natural language descriptions dataset used to train the `Narrations-2-SQL` is highlighted in Appendix D and the 5000 queries tested by `S-NAR` is provided in Appendix E.

A QUESTIONNAIRES

This appendix contains the questionnaires that were used in this study.

Generating Narrations from SQL Queries using Context-Free Grammars

Narrations provide explanations of a concept in a domain. For example, a program narrator explains details about lines of code. In this study, narrations are described as a textual explanation of a query. We have developed a tool called SQL Narrator that generates narrations for nested SQL queries. This section contains general questions about SQL queries. The next section presents specific questions on the SQL Narrator.

* Required

1. **Have you used a narrator before?** *

   Mark only one oval.

   - Yes
   - No
   - Maybe

2. **What was the narrator's task?**

   ____
   ____
   ____
   ____
   ____

3. **How would you rate the narrator on a scale from 1 to 10 (1 for not at all useful and 10 for very useful)** *

   Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Not at all useful											Very useful

4. **Are you familiar with simple SQL queries?** *

   Mark only one oval.

   - Yes
   - No
   - Maybe

5. **Are nested queries too difficult?** *

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

6. **Have you used a SQL-based Narrator before?** *

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

7. **Would you be interested in using a SQL-based narrator?** *

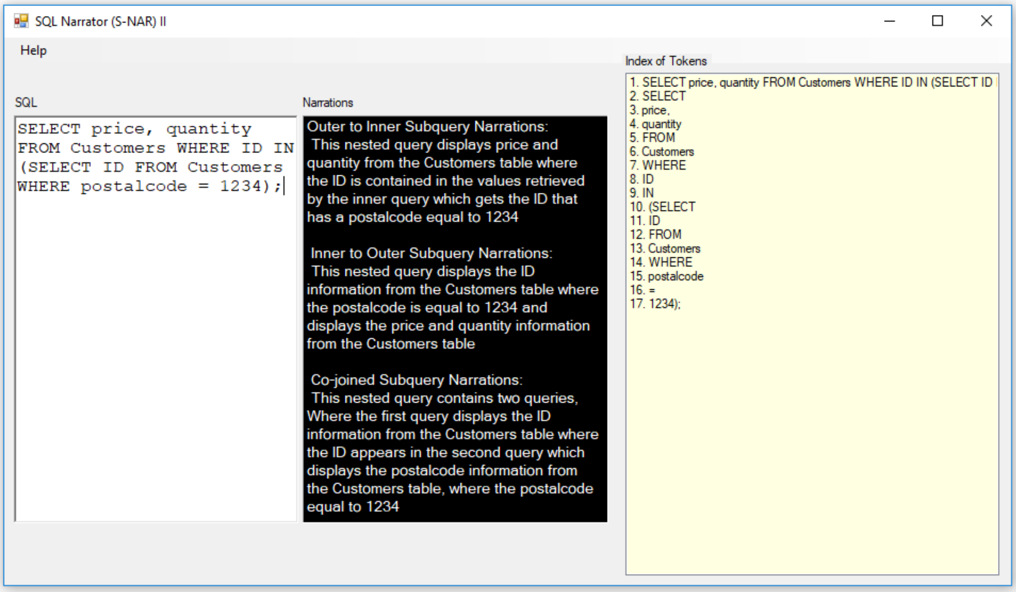
Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

SQL Narrator

Specific questions on the SQL Narrator tool. SQL Narrator generates narrations from nested SQL queries. Answer the following questions.

The SQL Narrator tool generates narrations from a nested SQL query



8. **Was the SQL Narrator easy to use?** *

Mark only one oval.

- Yes
- No
- Maybe

9. **Were you able to comprehend the nested queries using SQL Narrator?** *

Mark only one oval.

- Yes
- No
- Maybe

10. **Out of the three narrations generated, which one were you able to comprehend?** *

Mark only one oval.

- Inner to Outer Subquery Narration
- Outer to Inner Subquery Narration
- Co-joined Subquery Narration

11. **How would you rate the narrator on a scale of 1 to 10 (1 for not at all useful and 10 for very useful)?** *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Less useful											Very useful

12. **Any suggestions/comments to improve the SQL Narrator to aid your cognitive process?**

Synthesis of SQL Queries from Narrations

In this study, narrations are translated into SQL Queries. We have developed a tool that translates a natural language specification into SQL queries called narration-2-SQL. This section contains general questions on SQL queries. The next section presents specific questions about the narrations-2-SQL tool.

* Required

1. **Are you familiar with SQL?** *
   Mark only one oval.
   - Yes
   - No
   - Maybe

2. **Rate your familiarity with SQL on a scale of 1 to 10?** *
   Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Not at all familiar											Very familiar

3. **Are you familiar with the CREATE, SELECT, UPDATE and DELETE operations?** *
   Mark only one oval.
   - Yes
   - No
   - Maybe

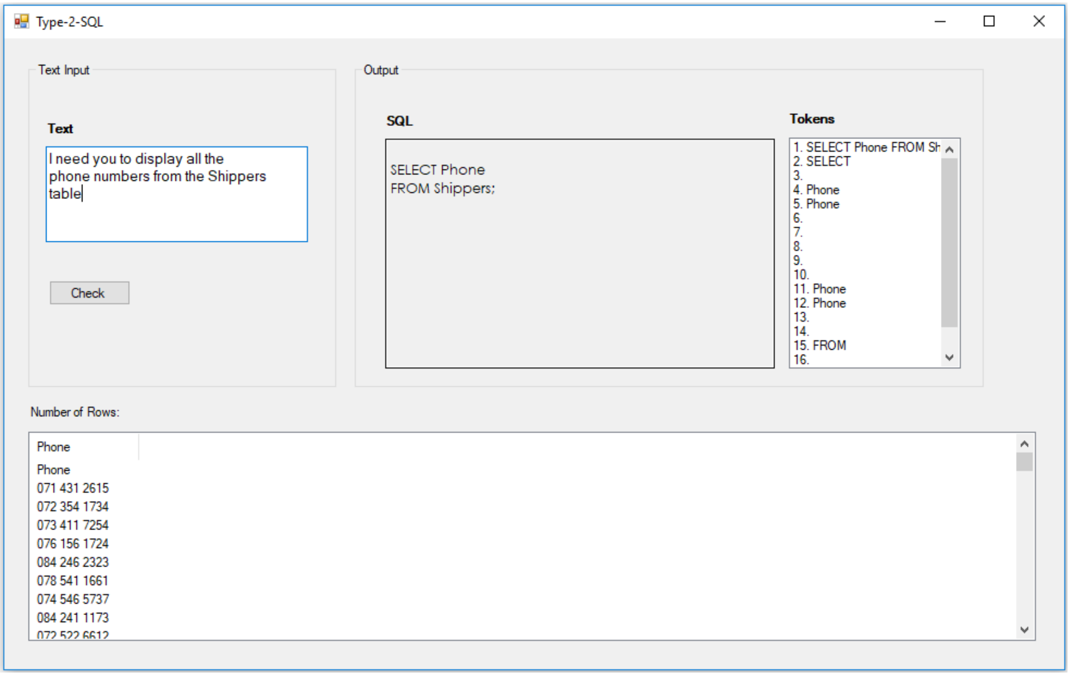
4. **Do you know of any tool that helps you to transform a natural language specification into SQL query before?** *
   Mark only one oval.
   - Yes
   - No
   - Maybe

5. **If yes, what is the name of such a tool?**

Narrations-2-SQL tool

Specific questions on the Narrators to SQL tool. This tool translates a natural language specification into SQL. Answer the following questions.

The tool translates natural language specifications into SQL queries



6. **Was Narration-2-SQL easy to use? ***

Mark only one oval.

- Yes
- No
- Maybe

7. **Was the generated SQL query a correct translation of your narration? ***

Mark only one oval.

- Yes
- No
- Maybe

8. **Do you think this tool will assist end users in the industry who has no knowledge of SQL to work with SQL? ***

Mark only one oval.

- Yes
- No
- Maybe

9. **How would you rate Narration-2-SQL (1 for not at all useful and 10 for very useful)?** *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Less useful											Very useful

10. **Any suggestions to improve Narration-2-SQL to aid your understanding?**

Powered by
Google Forms

Generating SQL Queries from Visual Specifications

This study proposes the use of interactive visualisation technique to aid the understanding of SQL. We have developed a tool that uses images that depicts SQL operation to generate SQL queries. This section contains general questions about the use of visualisers. The next section presents specific questions about the SQL visualiser that was developed for this study.

* Required

1. **Have you used a visualiser before?** *

Mark only one oval.

- Yes
- No
- Maybe

2. **What was the visualiser task?** *

3. **How would you rate the visualiser on a scale from 1 to 10** *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Not at all useful											Very useful

4. **Are you familiar with SQL** *

Mark only one oval.

- Yes
- No
- Maybe

5. **Rate your familiarity with SQL** *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Less familiar											Very familiar

6. **Have you used a SQL-based visualiser** *

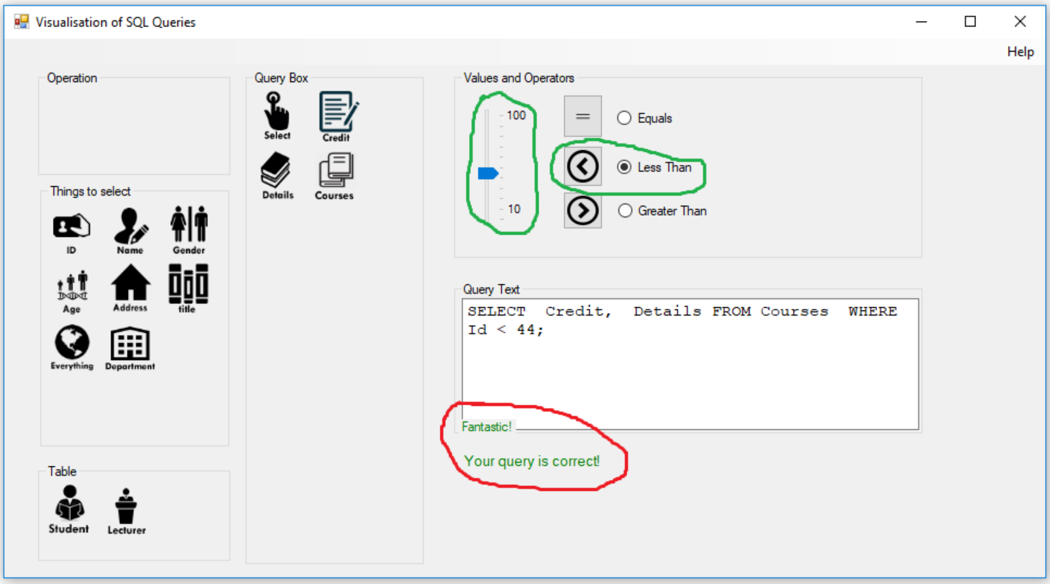
Mark only one oval.

- Yes
- No
- Maybe

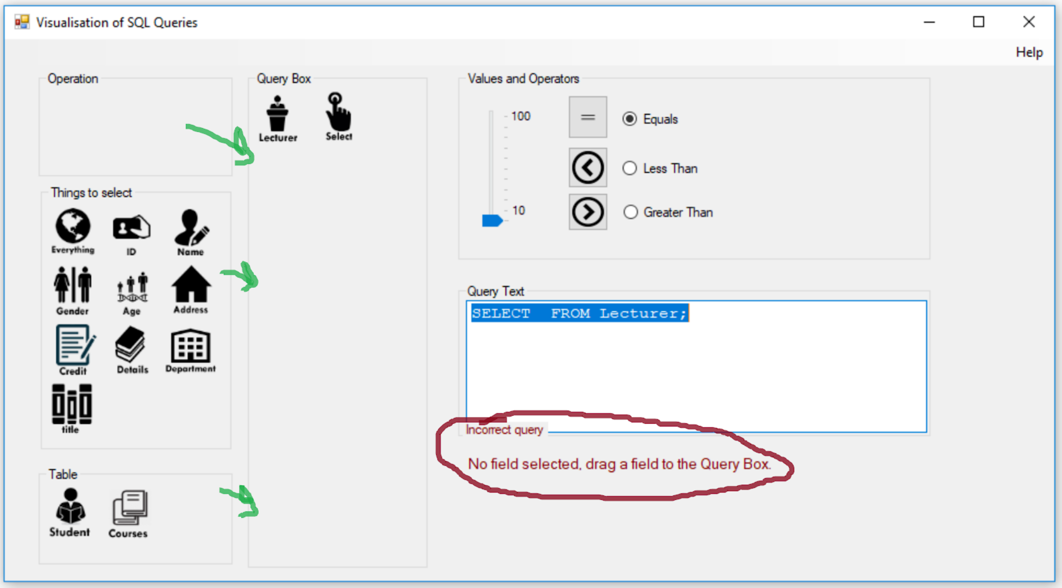
SQL Visualiser

Specific questions on SQL Visualiser. The SQL visualiser uses the drag and drop method to generate SQL queries.

The SQL Visualiser, generating a correct successful query



The SQL visualiser generating a wrong query



7. **Is the visualiser easy to use?** *
Mark only one oval.

- Yes
- No
- Maybe

8. **Were you able to synthesize the basic SQL queries?** *
Mark only one oval.

- Yes
- No
- Maybe

9. **Do icons help you analyse the syntax of SQL queries** *
Mark only one oval.

- Yes
- No
- Maybe

10. **How would you rate the visualiser on a scale of 1 to 10?** *
Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Less useful											Very useful

11. **Any suggestions/comments to improve the visualiser to aid your cognitive process?** *

Powered by Google Forms

Synthesis of SQL Queries from Verbal Specifications

In this study, a speech to SQL method was proposed. We have developed a tool called TalkSQL, that takes voice inputs from a user, converts these words into SQL queries and returns feedback to the user. This section contains general questions on users' familiarity with SQL and voice-based NLIDBs. The next section presents specific questions about the TalkSQL tool.

* Required

1. **Are you familiar with simple SQL queries?** *

   Mark only one oval.

   - Yes
   - No
   - Maybe

2. **Are you familiar with the CRUD (Create, Read, Update, Delete) operations?** *

   Mark only one oval.

   - Yes
   - No
   - Maybe

3. **Have you used a voice-based natural language interface to databases (NLIDB) before?** *

   Mark only one oval.

   - Yes
   - No
   - Maybe

4. **Would you be interested in a voice-based NLIDB?** *

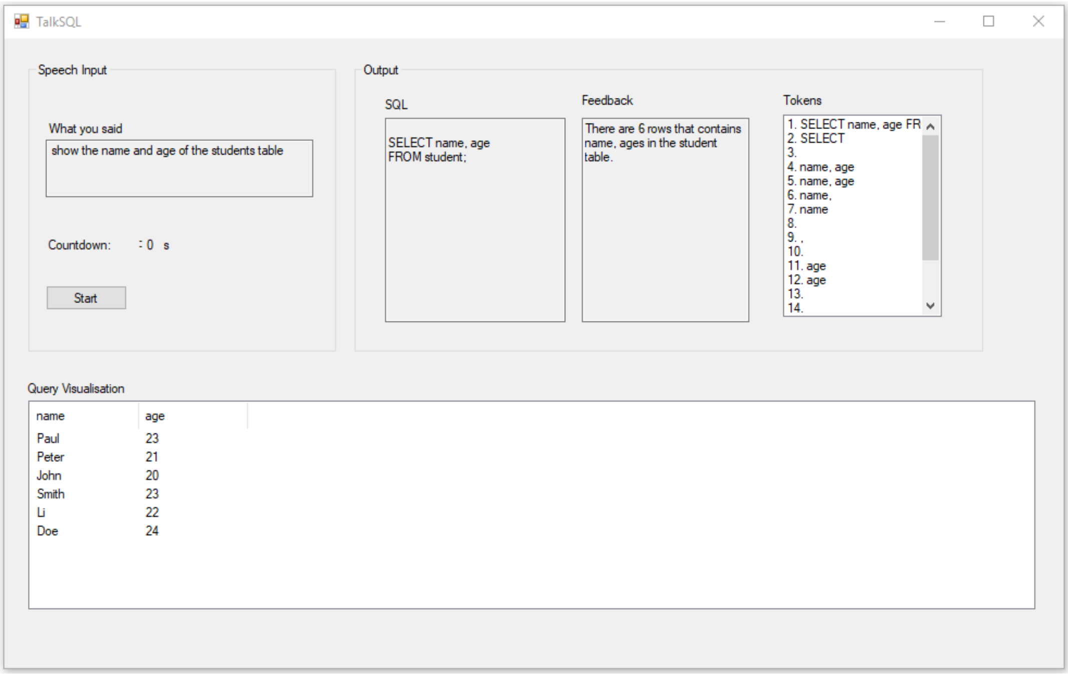
   Mark only one oval.

   - Yes
   - No
   - Maybe

TalkSQL

TalkSQL takes voice inputs from a user, convert these words into SQL queries and returns feedback to a user

The TalkSQL tool takes voice inputs from a user and generates a query.



5. **Was TalkSQL easy to use?** *

Mark only one oval.

- Yes
- No
- Maybe

6. **Were you able to understand the CRUD command using TalkSQL?** *

Mark only one oval.

- Yes
- No
- Maybe

7. **Do you think this might help visually impaired learners understand SQL?** *

Mark only one oval.

- Yes
- No
- Maybe

8. **Was the feedback you received comprehensive?** *

Mark only one oval.

- Yes
- No
- Maybe

9. **How would you rate TalkSQL (1 for not at all useful and 10 for very useful)?** *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Less useful											Very useful

10. **Any suggestions to improve TalkSQL to aid your understanding?**

B

RE LIBRARY

This section contains the REs library that have been specified in the .NET framework, written in VB.NET. This was used to recognise simple SQL query constructs as seen in Listing 20.

Listing 20: REs defined in .NET for SQL query constructs

```
'for all letters including hyphens
letter As String = "([A-Za-z-])"
s_spc As String = "(\s)"
n_spc As String = "(\s+)"
spc As String = "(\s*)"
all As String = "(\*)"
comma As String = "(\,)"
bra_open As String = "(\()"
bra_close As String = "(\))"
ass_sym As String = "(\=)"
greater_than As String = "(\>)"
less_than As String = "(\<)"
not_equal_to As String = "(\!\=)"
less_than_equal As String = "(\<\=)"
greater_than_equal As String = "(\>\=)"
not_greater_than As String = "(\!\>)"
not_less_than As String = "(\!\<)"
or_greater_or_less As String = "(\<\>)"
ident As String = "([A-Za-z_][A-Za-z0-9_]+)"
number As String = "[1-9][0-9]*"
semi_colon As String = "(\;)"
float_number As String = "(\d\d?\.\d\d?)"

val_in_quote As String = "((\')" & "(" & _ident & "|" & _number & "|" & _float_number & "|"
      & _n_spc & "|" & _comma & ")+" & "(\'))"

list_of_vals_in_quote As String = "(" & "(" & _val_in_quote & _spc & _comma & _spc & ")*(" &
      _val_in_quote & "))"

ident_sep_by_comma As String = "(" & "(" & _ident & _spc & _comma & _spc & ")*(" & _ident &
     "))"

list_of_values_sep_by_comma As String = _bra_open & _spc & _list_of_vals_in_quote & _spc &
     _bra_close

comp_op As String = "(" & _ass_sym & "|" & _greater_than & "|" & _less_than & "|" &
     _not_equal_to & "|" & _less_than_equal & "|" & _greater_than_equal & "|" &
     _not_less_than & "|" & _or_greater_or_less & ")"


'----------------- SQL (INSERT) ----------------------------
insert_suffix_into As String = _ident & _spc & _bra_open & _ident_sep_by_comma & _bra_close
insert_suffix_end As String = _bra_open & _list_of_values_sep_by_comma & _bra_close
insert_command As String = "(INSERT)" & _n_spc & "(INTO)" & _n_spc & _insert_suffix_into &
     _spc & "(VALUES)" & _spc & _insert_suffix_end & _semi_colon


'----------------- SQL (CREATE_TABLE) ----------------------------
create_suffix_into As String = _ident & _spc & _bra_open & _spc & _ident_sep_by_comma & _spc
      & _bra_close
create_command As String = "(CREATE)" & _n_spc & "(TABLE)" & _n_spc & _create_suffix_into &
     _semi_colon

'----------------- SQL (CREATE_DB) ----------------------------------------
create_database As String = "(CREATE)" & _n_spc & "(DATABASE)" & _n_spc & ("IF") & _n_spc &
     ("NOT") & _n_spc & ("EXISTS") & _n_spc & _ident & _semi_colon


'----------------- SQL (DROP) ----------------------------
drop_suffix_into As String = _ident_sep_by_comma
drop_command As String = "(DROP)" & _n_spc & "(DATABASE)" & _n_spc & "(IF)" & _spc & "(
     EXISTS)" & _spc & _drop_suffix_into & _semi_colon
drop_command_table As String = "(DROP)" & _n_spc & "(TABLE)" & _n_spc & "(IF)" & _spc & "(
     EXISTS)" & _spc & _drop_suffix_into & _semi_colon
```

```
'----------------- SQL (RENAME) ---------------------------
rename_suffix_into As String = _ident & _n_spc & "(TO)" & _n_spc & _ident
rename_command As String = "(RENAME)" & _n_spc & "(TABLE)" & _n_spc & _rename_suffix_into & _semi_colon

'----------------- SQL (TRUNCATE) ---------------------------
truncate_suffix_into As String = _ident
truncate_command As String = "(TRUNCATE)" & _n_spc & "(TABLE)" & _n_spc & _truncate_suffix_into & _semi_colon

'----------------- SQL (ALTER) ---------------------------
alter_suffix_into As String = _ident & _n_spc & "(RENAME)" & _n_spc & "(TO)" & _n_spc & _ident
alter_command As String = "(ALTER)" & _n_spc & "(TABLE)" & _n_spc & _alter_suffix_into & _semi_colon

'----------------- SQL (DELETE) ---------------------------
delete_suffix_into As String = _ident & _n_spc & "(WHERE)" & _n_spc & _ident & _comp_op & _val_in_quote
delete_command As String = "(DELETE)" & _n_spc & "(FROM)" & _n_spc & _delete_suffix_into & _semi_colon

'----------------- SQL (SELECT) ---------------------------
select_suffix_all As String = _all
select_suffix_one_or_more As String = "(" & _ident_sep_by_comma & "|" & "(\*)" & ")+"
select_numb_or_string_in_quote As String = "(" & "((\')" & _ident & "(\'))" & "|" & _number & ")+"
select_command_all As String = "(SELECT)" & _n_spc & _select_suffix_all & _n_spc & "(FROM)" & _n_spc & _ident & _semi_colon
select_command_all_more As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _semi_colon
select_command_distinct As String = "(SELECT)" & _n_spc & "(DISTINCT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _semi_colon
select_command_where As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(WHERE)" & _n_spc & _select_suffix_one_or_more & _comp_op & _select_numb_or_string_in_quote & _semi_colon
select_command_where_and As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(WHERE)" & _n_spc & _select_suffix_one_or_more & _comp_op & _select_numb_or_string_in_quote & _n_spc & "(AND)" & _n_spc & _select_suffix_one_or_more & _comp_op & _select_numb_or_string_in_quote & _semi_colon
select_command_where_or As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(WHERE)" & _n_spc & _select_suffix_one_or_more & _comp_op & _select_numb_or_string_in_quote & _n_spc & "(OR)" & _n_spc & _select_suffix_one_or_more & _comp_op & _select_numb_or_string_in_quote & _semi_colon
select_command_where_not As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(WHERE)" & _n_spc & "(NOT)" & _n_spc & _select_suffix_one_or_more & _comp_op & _select_numb_or_string_in_quote & _semi_colon
select_command_where_in As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(WHERE)" & _n_spc & _ident & _n_spc & "(IN)" & _n_spc & _list_of_values_sep_by_comma & _semi_colon
select_command_where_between As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(WHERE)" & _n_spc & _ident & _n_spc & "(BETWEEN)" & _n_spc & _select_numb_or_string_in_quote & _n_spc & "(AND)" & _n_spc & _select_numb_or_string_in_quote & _semi_colon
select_command_where_like As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(WHERE)" & _n_spc & _ident & _n_spc & "(LIKE)" & _n_spc & _select_numb_or_string_in_quote & _semi_colon
select_command_where_groupby As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(GROUP BY)" & _n_spc & _ident & _semi_colon
select_command_where_orderby_asc As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(ORDER BY)" & _n_spc & _ident & _n_spc & "(ASC)" & _semi_colon
select_command_where_orderby_desc As String = "(SELECT)" & _n_spc & _select_suffix_one_or_more & _n_spc & "(FROM)" & _n_spc & _ident & _n_spc & "(ORDER BY)" & _n_spc & _ident & _n_spc & "(DESC)" & _semi_colon
select_command_count As String = "(SELECT)" & _n_spc & "(COUNT)" & _bra_open & _ident & _bra_close & _n_spc & "(FROM)" & _n_spc & _ident & _semi_colon
```

C

CFG RULES

The following contains the attributed grammar (atg) of the SQL source language used by Coco/R to generate a scanner and parser for the language. This ideas were taken from the EBNF grammar defined by Ron Savage [Savage 2017]. Listing 21 shows the atg file used by the Coco/R engine.

Listing 21: Attributed grammar design using Coco/R

```
COMPILER SqlGrammar
public Narrator narrator;

CHARACTERS
letter     = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_".
digit      = "0123456789".

TOKENS
ident                   =       letter { letter | digit | '_' }.
number                  =       digit { digit }.
semi_colon              =       ';'.

IGNORE '\r' + '\n' + '\t'

PRODUCTIONS

SqlGrammar
                     (. narrator = new Narrator(); .)
=
TruncateCommand | DeleteCommand | SelectCommand .

TruncateCommand
=
                             (. string tableName; .)
"TRUNCATE" "TABLE" StringVal<out tableName> semi_colon  (. narrator.NarrateTruncate(
    tableName); .).

DeleteCommand
             (. ArrayList list = null; .)
=
"DELETE" { "*" | StringList<out list > } "FROM" StringVal<out string tableName> semi_colon
(. narrator.NarrateDelete(tableName, list); .).


SelectCommand
             (.      bool nested = false;
bool noConditions = false;
string comparisonOp = null;
string conditionColumn = null;
object criteriaValue = null;
string logicalOp = null;
ArrayList innerOptions = new ArrayList();
string innerTableName = null;
string innerConditionValue = null;
string innerComparisonOp = null; .)
=
"SELECT" StringList<out ArrayList options> "FROM" StringVal<out string tableName>
(. noConditions = true; .)
[
"WHERE" StringVal<out conditionColumn>
(
LogicalOp<out logicalOp>
"("
"SELECT" StringList<out innerOptions> "FROM" StringVal<out innerTableName>
"WHERE" StringVal<out innerConditionValue> ComparisonOp<out innerComparisonOp> AnyValue<out
    criteriaValue>
")"
             (. nested = true; noConditions = false;.)

|

ComparisonOp<out comparisonOp> AnyValue<out criteriaValue>
(. noConditions = false; .)
```

```
)
] semi_colon
             (. narrator.NarrateSelect(
nested,
noConditions,
options,
tableName,
comparisonOp,
conditionColumn,
innerOptions,
innerTableName,
innerConditionValue,
innerComparisonOp,
criteriaValue); .).


StringVal<out String s>
=
                             (. s = null;   .)
ident
                     (. s = t.val;   .).

StringList<out ArrayList list >
=
StringVal< out string s>                                                         (.
     list = new ArrayList{s}; .)
{"," StringVal< out s>                                                           (.
     list.Add(s); .)
}.

LogicalOp<out string op>                                                         (.
     op = null; .)
=
"AND"
                     (. op = t.val; .)
|
"OR"
                     (. op = t.val; .)
|
"IN"
                     (. op = t.val; .)
|
"NOT"
                     (. op = t.val; .)
|
"SOME"
                     (. op = t.val; .)
|
"ALL"
                     (. op = t.val; .).

ComparisonOp<out string op>                                                      (.
     op = null; .)
=
"="
                             (. op = "equal to"; .)
|
">"
                             (. op = "greater than"; .)
|
"<"
                             (. op = "less than"; .)
|
">="
                     (. op = "igreater than or equal to"; .)
|
"<="
                     (. op = "less than or equal to"; .)
|
"<>"
                     (. op = "not "; .).


AnyValue<out object value>                                                       (.
     value = null; .)
=
ident
                     (. value = "'" + t.val + "'"; .)
|
number
                     (. value = t.val; .).
```

```
//quote = "'" .
//string_num = quote | number.
//ident_list              =       ident   {"," ident}.
//logicalop               =       "AND" | "OR" | "IN" | "NOT" | "SOME" | "ALL".
END SqlGrammar.
```

D

NATURAL LANGUAGE QUERY TO SQL TRANSLATION

This following shows the JSON file that contains 204 translations of natural language descriptions into SQL queries (as shown in Listing 22). This dataset was used to train the `narrations-2-SQL` tool as discussed in Chapter 6. This is broken into *Item*, denoting the numbering, *Narrations* described as natural language descriptions, and the equivalent SQL queries. We have only presented 20 items here, the complete list can be accessed via: `http://tiny.cc/fe2oiz`.

Listing 22: JSON file containing natural language descriptions of SQL queries

```
[
{
"Item": 1,
"Narrations": "Please, show me all the information from the
    customers table.",
"SQL Queries": "SELECT * FROM Customers;"
},
{
"Item": 2,
"Narrations": "Retrieve all the order details information",
"SQL Queries": "SELECT * FROM order_details;"
},
{
"Item": 3,
"Narrations": "Display the orders information",
"SQL Queries": "SELECT * FROM orders;"
},
{
"Item": 4,
"Narrations": "Display all the products details",
"SQL Queries": "SELECT * FROM products;"
},
{
"Item": 5,
"Narrations": "Display all employee records",
"SQL Queries": "SELECT * FROM employee;"
},
{
"Item": 6,
"Narrations": "Display all the categories information",
"SQL Queries": "SELECT * FROM Categories;"
},
{
"Item": 7,
"Narrations": "Please can you show me all the shippers details
    from the table",
"SQL Queries": "SELECT * FROM shippers;"
},
{
"Item": 8,
```

```
"Narrations": "I need you to select all the suppliers data",
"SQL Queries": "SELECT * FROM suppliers;"
},
{
"Item": 9,
"Narrations": "Show all the employee cities.",
"SQL Queries": "SELECT cities FROM employees;"
},
{
"Item": 10,
"Narrations": "Show me only the employee countries.",
"SQL Queries": "SELECT country FROM employees;"
},
{
"Item": 11,
"Narrations": "Show all the employeeID.",
"SQL Queries": "SELECT employeeID FROM employees;"
},
{
"Item": 12,
"Narrations": "Select all ids from the customer table.",
"SQL Queries": "SELECT * FROM Customerdemographics;"
},
{
"Item": 13,
"Narrations": "List all customers from South Africa or USA",
"SQL Queries": "SELECT Id, FirstName, LastName, City, Country
   FROM Customers WHERE Country = 'South Africa' OR Country =
   'USA';"
},
{
"Item": 14,
"Narrations": "select the Customer Name and company Name",
"SQL Queries": "SELECT ContactName, CompanyName FROM Customers
   ;"
},
{
"Item": 15,
"Narrations": "select all columns from customer table  where
   the Country column has South Africa for its value",
"SQL Queries": "SELECT * FROM Customers WHERE Country='South
   Africa';"
},
{
"Item": 16,
"Narrations": "return only the Customer contact name and phone
   number where country is equal to South Africa",
"SQL Queries": "SELECT phone, ContactName FROM Customers WHERE
   Country='Sout-Africa';"
},
{
"Item": 17,
"Narrations": "select the First name and title from customers",
```

```
"SQL Queries": "SELECT ContactName, ContactTitle FROM Customers
    ;"
},
{
"Item": 18,
"Narrations": "List the first name, Phone, and city of all
    customers",
"SQL Queries": "SELECT ContactName, phone, city FROM Customers
    ;"
},
{
"Item": 19,
"Narrations": "List the order id, order date and shipped date
    for all orders.",
"SQL Queries": "SELECT orderID, orderDate and shippedDate FROM
    orders;"
},
{
"Item": 20,
"Narrations": "List the customers in Sweden",
"SQL Queries": "SELECT * FROM Customer WHERE Country = 'Sweden
    ';"
}
]
```

E DATASET OF SQL QUERIES

The following contains SQL queries scrapped from the Internet. In total, 5000 queries were scrapped from the Internet. We have only showed 44 queries here. The entire file can be accessed via http://tiny.cc/qs1adz. Listing 23 shows some of queries scrapped from Internet.

Listing 23: Dataset of SQL queries scrapped from the Internet

```sql
INSERT INTO Student (SELECT * FROM LateralStudent);
INSERT INTO Student (ROLL_NO,NAME,Age) SELECT ROLL_NO, NAME, Age FROM LateralStudent;
INSERT INTO Student SELECT * FROM LateralStudent WHERE Age = 18;
INSERT INTO categories(category_id, category_name)VALUES(150, 'Miscellaneous');
INSERT INTO customers(customer_id, last_name, first_name)SELECT employee_number AS customer_id, last_name, first_name FROM employees WHERE employee_number < 1003;
SELECT name(s)FROM student WHERE name = 'peter' AND name = 'doe';
SELECT name AS 'Alias' FROM student;
SELECT AVG(name)FROM student;
SELECT name(s)FROM student WHERE name BETWEEN 'peter' AND 'doe';
SELECT name,CASE WHEN condition THEN 'Result_1'WHEN condition THEN 'Result_2'ELSE 'Result_3'END FROM student;
SELECT COUNT(name)FROM student;
SELECT B.FirstName AS FirstName1, B.LastName AS LastName1, A.FirstName AS FirstName2, A.LastName AS LastName2, B.City, B.Country FROM Customer A, Customer B WHERE A.Id <> B.Id AND A.City = B.City AND A.Country = B.Country ORDER BY A.Country;
SELECT column-names FROM table-name UNION SELECT column-names FROM table-name;
SELECT 'Customer' As Type,FirstName + ' ' + LastName AS ContactName,City, Country, Phone FROM Customer UNION SELECT 'Supplier', ContactName, City, Country, Phone FROM Supplier;
SELECT column-names FROM table-name1 WHERE value IN (SELECT column-name FROM table-name2 WHERE condition);
SELECT column1 = (SELECT column-name FROM table-name WHERE condition),column-names FROM table-name WEHRE condition;
SELECT ProductName FROM Product WHERE Id IN (SELECT ProductId FROM OrderItem WHERE Quantity > 100);
SELECT FirstName, LastName, OrderCount = (SELECT COUNT(O.Id) FROM [Order] O WHERE O.CustomerId = C.Id) FROM Customer C ;
SELECT column-names FROM table-name WHERE column-name operator ANY (SELECT column-name FROM table-name WHERE condition);
SELECT column-names FROM table-name WHERE column-name operator ALL(SELECT column-name FROM table-name WHERE condition);
```

```
SELECT ProductName FROM Product WHERE Id = ANY(SELECT ProductId FROM OrderItem WHERE Quantity = 1);
SELECT DISTINCT FirstName + ' ' + LastName as CustomerName FROM Customer, [Order]WHERE Customer.Id = [Order].CustomerId AND TotalAmount > ALL (SELECT AVG(TotalAmount)FROM [Order] GROUP BY CustomerId);
SELECT column-names FROM table-name WHERE EXISTS (SELECT column-name FROM table-name WHERE condition);
SELECT CompanyName FROM Supplier WHERE EXISTS(SELECT ProductName FROM Product WHERE SupplierId = Supplier.Id AND UnitPrice > 100)        ;
SELECT column-names INTO new-table-name FROM table-name WHERE EXISTS(SELECT column-name FROM table-name WHERE condition);
SELECT * INTO SupplierUSA FROM Supplier WHERE Country = 'USA';
INSERT INTO table-name (column-names)SELECT column-names FROM table-name WHERE condition;
INSERT INTO Customer (FirstName, LastName, City, Country, Phone)SELECT LEFT(ContactName, CHARINDEX(' ',ContactName) - 1) AS FirstName,SUBSTRING(ContactName, CHARINDEX(' ', ContactName) + 1, 100) AS LastName,City, Country, Phone FROM Supplier WHERE Country = 'Canada';
SELECT column_list FROM table-name [WHERE Clause][GROUP BY clause][HAVING clause][ORDER BY clause];
SELECT first_name FROM student_details;
SELECT first_name, last_name FROM student_details;;
SELECT first_name + ' ' + last_name AS emp_name FROM employee;
SELECT * FROM EMPLOYEE_TBL;
SELECT EMP_ID FROM EMPLOYEE_TBL;
SELECT EMP_ID FROM EMPLOYEE_TBL;
SELECT EMP_ID, LAST_NAME FROM EMPLOYEE_TBL;
SELECT EMP_ID, LAST_NAME FROM EMPLOYEE_TBL WHERE EMP_ID = '333333333';
SELECT EMP_ID, LAST_NAME FROM EMPLOYEE_TBL WHERE CITY = 'INDIANAPOLIS' ORDER BY EMP_ID;
SELECT EMP_ID, LAST_NAME FROM EMPLOYEE_TBL WHERE CITY = 'INDIANAPOLIS' ORDER BY EMP_ID, LAST_NAME DESC;
SELECT EMP_ID, LAST_NAME FROM EMPLOYEE_TBL WHERE CITY = 'INDIANAPOLIS' ORDER BY 1;
INSERT INTO CUSTOMER(CustomerName,ContactName, Address, City, PostalCode, Country)('Cardinal','Tom B','Erichsen','Sagen 21','Stavanger','4006','Norway');
INSERT INTO CATEGORIES(Category_id, Category_Name)(150,'Miscellaneous');
INSERT INTO PRODUCT (ProductID, ProductName, Price, ProductDescription)(1,'Clamp',12.48,'Workbench clamp');
INSERT INTO CUSTOMER(FirstName,LastName,PhoneNumber, EmailAddress,priority,CreatedDate)('Jonah','Hook','0114022558','Jonahneverdull.com’,1,’2011-09-01’);
```

[Abadi *et al.* 2013] Daniel Abadi, Peter Boncz, Stavros Harizopoulos, Stratos Idreos, Samuel Madden, et al. The design and implementation of modern column-oriented database systems. *Foundations and Trends® in Databases*, 5(3):197–280, 2013.

[Abelló *et al.* 2008] Alberto Abelló, M Elena Rodríguez, Toni Urpí, Xavier Burgués, M José Casany, Carme Martín, and Carme Quer. LEARN-SQL: Automatic assessment of SQL based on IMS QTI specification. In *Eighth IEEE International Conference on Advanced Learning Technologies*, pages 592–593. IEEE, 2008.

[Abiteboul *et al.* 2005] Serge Abiteboul, Rakesh Agrawal, Phil Bernstein, Mike Carey, Stefano Ceri, Bruce Croft, David DeWitt, Mike Franklin, Hector Garcia Molina, Dieter Gawlick, et al. The Lowell database research self-assessment. *Communications of the ACM*, 48(5):111–118, 2005.

[Adams 2015] Nancy E Adams. Bloom's taxonomy of cognitive learning objectives. *Journal of the Medical Library Association: JMLA*, 103(3):152, 2015.

[Adams 2017] Marilyn Jager Adams. Failures to comprehend and levels of processing. *Theoretical issues in reading comprehension: Perspectives from cognitive psychology, linguistics, artificial intelligence and education*, 11:11, 2017.

[Ade-Ibijola and Obaido 2017] Abejide Ade-Ibijola and George Obaido. S-NAR: generating narrations of SQL queries using regular expressions. In *the ACM proceedings of Proceedings of the South African Institute of Computer Scientists and Information Technologists*, pages 11–18, 2017. Bloemfontein, Free State. ISBN: 978-1-4503-5250-5. https://dl.acm.org/citation.cfm?doid=3129416.3129454. [South Africa].

[Ade-Ibijola and Obaido 2019] Abejide Ade-Ibijola and George Obaido. XNorthwind: Grammar-driven synthesis of large datasets for DB applications. In *In International Journal of Computer Science (Scopus)*, pages 541–551, 2019. International Association of Engineers (IAENG). http://www.iaeng.org/IJCS/issues_v46/issue_4/IJCS_46_4_05.pdf. [Hong Kong].

[Ade-Ibijola *et al.* 2014] Abejide Ade-Ibijola, Sigrid Ewert, and Ian Sanders. Abstracting and narrating novice programs using regular expressions. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT), Centurion*, pages 19–28, 2014. ACM International Conference Series, doi: 10.1145/2664591.2664601, http://dl.acm.org/citation.cfm?id=2664601 [South Africa].

[Ade-Ibijola *et al.* 2015] Abejide Ade-Ibijola, Sigrid Ewert, and Ian Sanders. Introducing Code Adviser: A DFA-driven electronic programming tutor. In *Implementation and Application of Automata, Umeå,*, pages 307–312, 2015. Springer LNCS 9223, doi: 10.1007/978-3-319-22360-525, http://link.springer.com/chapter/10.1007/978-3-319-22360-5_25 [Sweden].

[Ade-Ibijola 2016a] Abejide Ade-Ibijola. FINCHAN: A grammar-based tool for automatic comprehension of financial instant messages. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT)*, pages 1–10, 2016. Johannesburg. ACM International Conference Series, ISBN: 978-1-4503-4805-8, http://dl.acm.org/citation.cfm?id=2987518, doi: http://dx.doi.org/10.1145/2987491.2987518 [South Africa].

[Ade-Ibijola 2016b] Abejide Olu Ade-Ibijola. *Automatic novice program comprehension for semantic bug detection*, 2016. A research thesis submitted for the award of Doctor of Philosophy in Computer Science, WITS [South Africa]. Available online: http://wiredspace.wits.ac.za/handle/10539/21019.

[Ade-Ibijola 2017a] Abejide Ade-Ibijola. Automata-aided estimation of similarity in novice programs. In *the proceedings of 46th Annual South African Computer Lecturers' Association (SACLA),*, pages 23–38, 2017. Magaliesburg, North-west, ISBN: 978-1-86822-846-1. http://hdl.handle.net/10210/241253 [South Africa].

[Ade-Ibijola 2017b] Abejide Ade-Ibijola. *New finite automata applications in novice program comprehension*. 2017. ISBN: 978-3-330-02905-7, LAP Lambert Academic Publishing, Bahnhofstrasse 28, Saarbrücken, URL: https://www.amazon.com/Finite-Automata-Applications-Program-Comprehension/dp/3330029056 [Germany].

[Ade-Ibijola 2017c] Abejide Ade-Ibijola. Synthesis of social media profiles using a probabilistic context-free grammar. In *the proceedings of the IEEE Conference of the Pattern Recognition Association of South Africa, Robotics, and Mechatronics (PRASA-RobMech)*, pages 104–109, 2017. Bloemfontein. ISBN: 978-1-5386-2313-8/17. http://ieeexplore.ieee.org/document/8261131/ [South Africa].

[Ade-Ibijola 2018a] Abejide Ade-Ibijola. Syntactic generation of practice novice programs in Python. In *Communications in Computer and Information Science (CCIS)*, volume 963, pages 158–172, 2018. Springer, Cham (Scopus). ISBN: 978-3-030-05813-5. https://link.springer.com/chapter/10.1007/978-3-030-05813-5_11 [Switzerland].

[Ade-Ibijola 2018b] Abejide Ade-Ibijola. Synthesis of regular expression problems and solutions. *In International Journal of Computers and Applications (Scopus), Taylor's and Francis.*, pages 1–17, 2018. URL: www.tandfonline.com/doi/full/10.1080/1206212X.2018.1482398 [United Kingdom].

[Ade-Ibijola 2019] Abejide Ade-Ibijola. Synthesis of hypothetical sociograms for social network analysis. In *5th IEEE International Conference on Soft Computing & Machine Intelligence (ISCMI), Nairobi*, pages 79–83, 2019. URL: https://ieeexplore.ieee.org/document/8703221. [Kenya].

[Affolter *et al.* 2019] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. A comparative survey of recent natural language interfaces for databases. *arXiv preprint arXiv:1906.08990*, 2019.

[Agarwal and Kumar 2016] Deepak Kumar Agarwal and Rahul Kumar. Spam filtering using SVM with different kernel functions. *International Journal of Computer Applications*, 136(5):0975–8887, 2016.

[Ahadi *et al.* 2015] Alireza Ahadi, Julia Prior, Vahid Behbood, and Raymond Lister. A quantitative study of the relative difficulty for novices of writing seven different types of SQL queries. In *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education*, pages 201–206. ACM, 2015.

[Ahadi *et al.* 2016] Alireza Ahadi, Vahid Behbood, Arto Vihavainen, Julia Prior, and Raymond Lister. Students' syntactic mistakes in writing seven different types of SQL queries and its application to predicting students' success. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 401–406. ACM, 2016.

[Aho *et al.* 1986] Alfred V Aho, Ravi Sethi, and Jeffrey D Ullman. Compilers, principles, techniques. *Addison Wesley*, 7(8):9, 1986.

[Aken and Michalisin 2007] Andrew Aken and Michael D Michalisin. The impact of the skills gap on the recruitment of MIS graduates. In *Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce*, pages 105–111. ACM, 2007.

[Al Omran and Treude 2017] Fouad Nasser A Al Omran and Christoph Treude. Choosing an NLP library for analyzing software documentation: a systematic literature review and a series of experiments. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 187–197. IEEE Press, 2017.

[Al-Radaideh and Al-Abrat 2019] Qasem A Al-Radaideh and Mohammed A Al-Abrat. An Arabic text categorization approach using term weighting and multiple reducts. *Soft Computing*, 23(14):5849–5863, 2019.

[Al-Shuaily and Renaud 2010] Huda Al-Shuaily and Karen Renaud. SQL patterns-a new approach for teaching SQL. In *8th HEA Workshop on Teaching, Learning and Assessment of Databases, Abertay-Dundee*, pages 29–40, 2010.

[Al Shuaily and Renaud 2016a] Huda Al Shuaily and Karen Renaud. A framework for SQL learning: Linking learning taxonomy, cognitive model and cross cutting factors. *World Academy of Science, Engineering and Technology, International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 10(9):3091–3097, 2016.

[Al Shuaily and Renaud 2016b] Huda Al Shuaily and Karen Renaud. A model of SQL learning. 2016.

[Al-Shuaily 2013] Huda Al-Shuaily. *SQL pattern design, development & evaluation of its efficacy*. PhD thesis, University of Glasgow, 2013.

[Aldahdooh and Naser 2017] Rami Aldahdooh and Samy S Abu Naser. Development and evaluation of the Oracle Intelligent Tutoring System. 2017.

[Alexander 1977] Christopher Alexander. *A pattern language: towns, buildings, construction*. Oxford university press, 1977.

[Allenstein *et al.* 2008] Brett Allenstein, Andrew Yost, Paul Wagner, and Joline Morrison. A query simulation system to illustrate database query execution. *ACM Special Interest Group on Computer Science Education Bulletin*, 40(1):493–497, 2008.

[Alsajjan and Dennis 2010] Bander Alsajjan and Charles Dennis. Internet banking acceptance model: Cross-market examination. *Journal of Business Research*, 63(9):957–963, 2010.

[Alsohybe *et al.* 2017] Nabeel T Alsohybe, Neama Abdulaziz Dahan, and Fadl Mutaher Ba-Alwi. Machine-translation history and evolution: Survey for Arabic-English translations. *arXiv preprint arXiv:1709.04685*, 2017.

[Amodei *et al.* 2016] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[Andreeva *et al.* 2018] Ol'ga Vyacheslavovna Andreeva, Mirabbas Bakhtiyarovich Bagirov, Anastasiya Aleksandrovna Dankina, Tat'yana Olegovna Fedorova, and Mariya Mikhailovna Sheveleva. Intellectual analysis of data on the basis of Stanford coreNLP for PoS tagging of texts in the Russian language. *Sistemy i Sredstva Informatiki [Systems and Means of Informatics]*, 28(2):145–153, 2018.

[Angeli *et al.* 2014] Gabor Angeli, Sonal Gupta, Melvin Johnson Premkumar, Christopher D Manning, Christopher Ré, Julie Tibshirani, Jean Y Wu, Sen Wu, and Ce Zhang. Stanford's distantly supervised slot filling systems for kbp 2014. In *Proceedings of the Seventh Text Analysis Conference*, 2014.

[Arasu *et al.* 2016] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. Stream: The Stanford Data Stream Management System. In *Data Stream Management*, pages 317–336. Springer, 2016.

[Ardito *et al.* 2014] Carmelo Ardito, Maria Francesca Costabile, and Hans-Christian Jetter. Gestures that people can understand and use. *Journal of Visual Languages & Computing*, 25(5):572–576, 2014.

[Armbrust *et al.* 2015] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. Spark SQL: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1383–1394. ACM, 2015.

[Artetxe *et al.* 2017] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017.

[Atchariyachanvanich *et al.* 2017] Kanokwan Atchariyachanvanich, Srinual Nalintippayawong, and Tanasab Permpool. Development of a mySQL sandbox for processing SQL statements: Case of DML and DDL statements. In *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–6. IEEE, 2017.

[Athenikos and Han 2010] Sofia J Athenikos and Hyoil Han. Biomedical question answering: A survey. *Computer methods and programs in biomedicine*, 99(1):1–24, 2010.

[Auria and Moro 2008] Laura Auria and Rouslan A Moro. Support vector machines (SVM) as a technique for solvency analysis. 2008.

[Awad and Khanna 2015] Mariette Awad and Rahul Khanna. *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. Apress, 2015.

[Azer *et al.* 2013] Samy A Azer, Michelle Mclean, Hirotaka Onishi, Masami Tagawa, and Albert Scherpbier. Cracks in problem-based learning: What is your action plan? *Medical teacher*, 35(10):806–814, 2013.

[Bahdanau *et al.* 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[Bahdanau *et al.* 2016] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

[Baik *et al.* 2019] Christopher Baik, HV Jagadish, and Yunyao Li. Bridging the semantic gap with SQL query logs in natural language interfaces to databases. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 374–385. IEEE, 2019.

[Baldridge 2005] Jason Baldridge. The openNLP project. *URL: http://opennlp. apache. org/index. html,(accessed 2 February 2012)*, page 1, 2005.

[Barber *et al.* 2018] Mark H Barber, Carsten Hagemann, and Christopher J Hockings. *Similar email spam detection*, 2018. US Patent App. 15/270,237.

[Barrera and Pachitariu 2018] Jorge Barrera and George Pachitariu. Big data: What is it? and is my data big enough? 2018.

[Barron *et al.* 2015] Andrew B Barron, Eileen A Hebets, Thomas A Cleland, Courtney L Fitzpatrick, Mark E Hauber, and Jeffrey R Stevens. Embracing multiple definitions of learning. *Trends in neurosciences*, 38(7):405–407, 2015.

[Bast and Haussmann 2015] Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440. ACM, 2015.

[Bastani *et al.* 2017] Osbert Bastani, Rahul Sharma, Alex Aiken, and Percy Liang. Synthesizing program input grammars. In *ACM Special Interest Group on Programming Languages*, volume 52, pages 95–110. ACM, 2017.

[Bateman and Zock 2003] John Bateman and Michael Zock. Natural language generation. In *The Oxford Handbook of Computational Linguistics 2nd edition*. 2003.

[Bates 1995] Madeleine Bates. Models of natural language understanding. *Proceedings of the National Academy of Sciences*, 92(22):9977–9982, 1995.

[Batini *et al.* 1991] Carlo Batini, Tiziana Catarci, Maria Francesca Costabile, and Stefano Levialdi. Visual query systems: A taxonomy. In *VDB*, pages 153–168, 1991.

[Batra 2018] Rahul Batra. A history of SQL and relational databases. In *SQL Primer*, pages 183–187. Springer, 2018.

[Bau 2015] David Bau. Droplet, a blocks-based editor for text code. *Journal of Computing Sciences in Colleges*, 30(6):138–144, 2015.

[Baykasoğlu *et al.* 2018] Adil Baykasoğlu, Burcu K Özbel, Nurhan Dudaklı, Kemal Subulan, and Mümin Emre Şenol. Process mining based approach to performance evaluation in computer-aided examinations. *Computer Applications in Engineering Education*, 26(5):1841–1861, 2018.

[Beaulieu 2009] Alan Beaulieu. *Learning SQL: Master SQL Fundamentals*. " O'Reilly Media, Inc.", 2009.

[Behari *et al.* 2016] Suren Behari, Aileen Cater-Steel, and Jeffrey Soar. Data science and big data analytics in financial services: A case study. *Handbook of Research on Driving Competitive Advantage through Sustainable, Lean, and Disruptive Innovation*, page 396, 2016.

[Belinkov *et al.* 2015] Yonatan Belinkov, Mitra Mohtarami, Scott Cyphers, and James Glass. Vectorslu: A continuous word vector approach to answer selection in community question answering systems. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 282–287, 2015.

[Ben-Hur and Weston 2010] Asa Ben-Hur and Jason Weston. A user's guide to support vector machines. In *Data mining techniques for the life sciences*, pages 223–239. Springer, 2010.

[Bengio and Senécal 2008] Yoshua Bengio and Jean-Sébastien Senécal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722, 2008.

[Bengio *et al.* 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[Bentley 1999] Jon Bentley. *Programming Pearls*. Addison–Wesley, Boston, MA, USA, 2nd edition, 1999.

[Berant and Liang 2014] Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, 2014.

[Bergin *et al.* 2012] Joseph Bergin, Jutta Eckstein, M Manns, H Sharp, K Marquardt, Jane Chandler, M Sipos, M Volter, and E Wallingford. *Pedagogical patterns: advice for educators*. CreateSpace, 2012.

[Berque *et al.* 2003] Dave Berque, Terri Bonebright, Seth Kinnett, Nathan Nichols, and Adam Peters. A case study in the design of software that uses auditory cues to help low vision students view notes on a blackboard. Georgia Institute of Technology, 2003.

[Berthold *et al.* 2009] Kirsten Berthold, Tessa HS Eysink, and Alexander Renkl. Assisting self-explanation prompts are more effective than open prompts when learning with multiple representations. *Instructional Science*, 37(4):345–363, 2009.

[Bian *et al.* 2014] Jiang Bian, Bin Gao, and Tie-Yan Liu. Knowledge-powered deep learning for word embedding. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 132–148. Springer, 2014.

[Bider and Rogers 2016] Ilia Bider and David Rogers. YASQLT–Yet Another SQL Tutor. In *International Conference on Conceptual Modeling*, pages 197–206. Springer, 2016.

[Bilgin 2019] Metin Bilgin. A study on named entity recognition with openNLP at English texts. *Journal of Applied Intelligent System*, 4(1):1–8, 2019.

[Bird *et al.* 2008] Steven Bird, Ewan Klein, Edward Loper, and Jason Baldridge. Multidisciplinary instruction with the natural language toolkit. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pages 62–70. Association for Computational Linguistics, 2008.

[Bird *et al.* 2009] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[Bird 2002] Steven Bird. Computational phonology. *arXiv preprint cs/0204023*, 2002.

[Bobrow *et al.* 1977] Daniel G Bobrow, Ronald M Kaplan, Martin Kay, Donald A Norman, Henry Thompson, and Terry Winograd. Gus, a frame-driven dialog system. *Artificial intelligence*, 8(2):155–173, 1977.

[Bocklisch *et al.* 2017] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*, 2017.

[Bogard 2004] Travis A Bogard. Instant messaging via telephone interfaces. *Google Patents*, 2004.

[Bondielli *et al.* 2018] Alessandro Bondielli, Lucia C Passaro, and Alessandro Lenci. CoreNLP-it: A UD pipeline for italian based on Stanford coreNLP. In *CLiC-it*, 2018.

[Bonham-Carter 2014] Graeme F Bonham-Carter. *Geographic information systems for geoscientists: modelling with GIS*, volume 13. Elsevier, 2014.

[Borchers and Thomas 2001] Jan O Borchers and John C Thomas. Patterns: what's in it for HCI? In *CHI'01 extended abstracts on Human factors in computing systems*, pages 225–226. ACM, 2001.

[Borchers 2000] Jan O Borchers. Interaction design patterns: twelve theses. In *Workshop, The Hague*, volume 2, page 3. Citeseer, 2000.

[Bornat 1987] Richard Bornat. *Programming from First Principles*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1987.

[Borodin *et al.* 2016] Andrey Borodin, Yuri Kiselev, Sergey Mirvoda, and Sergey Porshnev. Development of data aggregation capabilities in domain-specific query language for metallurgy. In *2016 Dynamics of Systems, Mechanisms and Machines (Dynamics)*, pages 1–6. IEEE, 2016.

[Borrego 2007] Maura Borrego. Development of engineering education as a rigorous discipline: A study of the publication patterns of four coalitions. *Journal of Engineering Education*, 96(1):5–18, 2007.

[Boser *et al.* 2003] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 2003.

[Bowden *et al.* 2019] Kevin K Bowden, Shereen Oraby, Amita Misra, Jiaqi Wu, Stephanie Lukin, and Marilyn Walker. Data-driven dialogue systems for social agents. In *Advanced Social Interaction with Agents*, pages 53–56. Springer, 2019.

[Bower 2008] Matt Bower. A taxonomy of task types in computing. In *ACM SIGCSE Bulletin*, volume 40, pages 281–285. ACM, 2008.

[Bozzelli *et al.* 2015] Laura Bozzelli, Bastien Maubert, and Sophie Pinchinat. Uniform strategies, rational relations and jumping automata. *Information and Computation*, 242:80–107, 2015.

[Bringhurst 2013] Robert Bringhurst. *The Elements of Typographic Style*. Version 4.0: 20th Anniversary Edition. Hartley & Marks Publishers, Point Roberts, WA, USA, 2013.

[Broadhurst and Trivedi 2018] Roderic Broadhurst and Harshit Trivedi. Malware in spam email: Trends in the 2016 australian spam intelligence data. *Available at SSRN 3413442*, 2018.

[Brodie 2014] Karin Brodie. Learning about learner errors in professional learning communities. *Educational Studies in Mathematics*, 85(2):221–239, 2014.

[Brooks 1977] Ruven Brooks. Towards a theory of the cognitive processes in computer programming. *International Journal of Man-Machine Studies*, 9(6):737–751, 1977.

[Brown 1998] C Marlin Brown. *Human-computer interface design guidelines*. Intellect Books, 1998.

[Brüggemann-Klein and Wood 1998] Anne Brüggemann-Klein and Derick Wood. One-unambiguous regular languages. *Information and Computation*, 142(2):182–206, 1998.

[Bruner and others 1966] Jerome Seymour Bruner et al. *Toward a theory of instruction*, volume 59. Harvard University Press, 1966.

[Brusilovsky *et al.* 2008] Peter Brusilovsky, Sergey Sosnovsky, Danielle H Lee, Michael Yudelson, Vladimir Zadorozhny, and Xin Zhou. An open integrated exploratorium for database courses. In *ACM Special Interest Group on Computer Science Education Bulletin*, volume 40, pages 22–26. ACM, 2008.

[Bui and Zeng-Treitler 2014] Duy Duc An Bui and Qing Zeng-Treitler. Learning regular expressions for clinical text classification. *Journal of the American Medical Informatics Association*, 21(5):850–857, 2014.

[Burileanu 2008] Dragos Burileanu. Spoken language interfaces for embedded applications. In *Human factors and voice interactive systems*, pages 135–161. Springer, 2008.

[Burtsev *et al.* 2018] Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, et al. Deeppavlov: open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, 2018.

[Cagliero *et al.* 2018] Luca Cagliero, Luigi De Russis, Laura Farinetti, and Teodoro Montanaro. Improving the effectiveness of SQL learning practice: a data-driven approach. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 980–989. IEEE, 2018.

[Cai *et al.* 2017] Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, Zijian Li, and Zhihao Liang. An encoder-decoder framework translating natural language to database queries. *arXiv preprint arXiv:1711.06061*, 2017.

[Caldeira 2008] Carlos Pampulim Caldeira. Teaching SQL: a case study. In *ACM Special Interest Group on Computer Science Education Bulletin*, volume 40, pages 340–340. ACM, 2008.

[Cambria and White 2014] Erik Cambria and Bebo White. Jumping NLP curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57, 2014.

[Cambria 2016] Erik Cambria. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107, 2016.

[Câmpeanu *et al.* 2003] Cezar Câmpeanu, Kai Salomaa, and Sheng Yu. A formal study of practical regular expressions. *International Journal of Foundations of Computer Science*, 14(06):1007–1018, 2003.

[Cantador *et al.* 2011] Ivan Cantador, Peter L Brusilovsky, and Tsvi Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). 2011.

[Cappel 2002] James J Cappel. Entry-level is job skills: A survey of employers. *Journal of Computer Information Systems*, 42(2):76–82, 2002.

[Cappers and van Wijk 2017] Bram CM Cappers and Jarke J van Wijk. Exploring multivariate event sequences using rules, aggregations, and selections. *IEEE transactions on visualization and computer graphics*, 24(1):532–541, 2017.

[Cassidy 2004] Simon Cassidy. Learning styles: An overview of theories, models, and measures. *Educational psychology*, 24(4):419–444, 2004.

[Catarci *et al.* 1997] Tiziana Catarci, Maria F Costabile, Stefano Levialdi, and Carlo Batini. Visual query systems for databases: A survey. *Journal of Visual Languages & Computing*, 8(2):215–260, 1997.

[Ceballos *et al.* 2012] Ariel Gregorio Ceballos, Michael Anthony Caruso, and Kenneth Klaus. *Criteria builder for query builder*, November 13 2012. US Patent 8,312,038.

[Cebollero *et al.* 2015] Miguel Cebollero, Jay Natarajan, and Michael Coles. Error handling and dynamic SQL. In *Pro T-SQL Programmer's Guide*, pages 589–612. Springer, 2015.

[Cegielski and Jones-Farmer 2016] Casey G Cegielski and L Allison Jones-Farmer. Knowledge, skills, and abilities for entry-level business analytics positions: A multi-method study. *Decision Sciences Journal of Innovative Education*, 14(1):91–118, 2016.

[Cembalo *et al.* 2011] Maurizio Cembalo, Alfredo De Santis, and Umberto Ferraro Petrillo. SAVI: a new system for advanced SQL visualization. In *Proceedings of the 2011 conference on Information technology education*, pages 165–170. ACM, 2011.

[Cereda and Neto 2017] Paulo Roberto Massa Cereda and João José Neto. Instrumenting a context-free language recognizer. In *International Conference on Enterprise Information Systems*, pages 203–210, 2017.

[Chaiklin 2003] Seth Chaiklin. The zone of proximal development in Vygotsky's analysis of learning and instruction. *Vygotsky's educational theory in cultural context*, 1:39–64, 2003.

[Chakrabarti 2004] Soumen Chakrabarti. Breaking through the syntax barrier: Searching with entities and relations. In *European Conference on Machine Learning*, pages 9–16. Springer, 2004.

[Chamberlin 2012] Donald D Chamberlin. Early history of SQL. *IEEE Annals of the History of Computing*, 34(4):78–82, 2012.

[Chandarana *et al.* 2017] Dharmil Chandarana, Vraj Shah, Arun Kumar, and Lawrence Saul. SpeakQL: towards speech-driven multi-modal querying. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 11. ACM, 2017.

[Chandra and Suaib 2014] Ashish Chandra and Mohammad Suaib. A survey on web spam and spam 2.0. *International Journal of Advanced Computer Research*, 4(2):634, 2014.

[Chao 2016] Po-Yao Chao. Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 95:202–215, 2016.

[Chaturvedi *et al.* 2018] Iti Chaturvedi, Erik Cambria, Roy E Welsch, and Francisco Herrera. Distinguishing between facts and opinions for sentiment analysis: Survey and challenges. *Information Fusion*, 44:65–77, 2018.

[Chen and Chen 2015] Yu-Ren Chen and Hsin-Hsi Chen. Opinion spam detection in web forum: a real case study. In *Proceedings of the 24th International Conference on World Wide Web*, pages 173–183. International World Wide Web Conferences Steering Committee, 2015.

[Chen *et al.* 2017a] Xingliang Chen, Antonija Mitrovic, and Moffat Mathews. How much learning support should be provided to novices and advanced students? In *IEEE 17th International Conference on Advanced Learning Technologies*, pages 39–43. IEEE, 2017.

[Chen *et al.* 2017b] Yun-Nung Chen, Asli Celikyilmaz, and Dilek Hakkani-Tür. Deep learning for dialogue systems. In *Proceedings of ACL 2017, Tutorial Abstracts*, pages 8–14, 2017.

[Cherpas 1992] Chris Cherpas. Natural language processing, pragmatics, and verbal behavior. *The Analysis of verbal behavior*, 10(1):135–147, 1992.

[Cheung *et al.* 2016] Alvin Cheung, Armando Solar-Lezama, et al. Computer-assisted query formulation. *Foundations and Trends® in Programming Languages*, 3(1):1–94, 2016.

[Chiang *et al.* 2012] Roger HL Chiang, Paulo Goes, and Edward A Stohr. Business intelligence and analytics education, and program development: A unique opportunity for the information systems discipline. *ACM Transactions on Management Information Systems (TMIS)*, 3(3):12, 2012.

[Ching *et al.* 2018] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[Chomsky and Lightfoot 2002] Noam Chomsky and David W Lightfoot. *Syntactic structures*. Walter de Gruyter, 2002.

[Chomsky 1956] Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

[Chomsky 1959] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.

[Chong *et al.* 2017] Frederic T Chong, Diana Franklin, and Margaret Martonosi. Programming languages and compiler design for realistic quantum hardware. *Nature*, 549(7671):180, 2017.

[Chou and Chou 2000] David C Chou and Amy Y Chou. A guide to the Internet revolution in banking. *Information Systems Management*, 17(2):51–57, 2000.

[Chowdhury 2003] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[Christina *et al.* 2010] V Christina, S Karpagavalli, and G Suganya. A study on email spam filtering techniques. *International Journal of Computer Applications*, 12(1):7–9, 2010.

[Chu *et al.* 2017] Shumo Chu, Chenglong Wang, Konstantin Weitz, and Alvin Cheung. Cosette: An automated prover for SQL. In *Conference on Innovative Data Systems Research*, 2017.

[Church and de Oliveira 2013] Karen Church and Rodrigo de Oliveira. What's up with Whatsapp?: comparing mobile instant messaging behaviors with traditional SMS. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, pages 352–361. ACM, 2013.

[Churches 2010] Andrew Churches. *Bloom's digital taxonomy*, 2010.

[Cienciala *et al.* 2014] Ludec Cienciala, Lucie Ciencialová, and Erzsébet Csuhaj Varjú. Towards p colonies processing strings. *Proceedings of the Twelfth Brainstorming Week on Membrane Computing, 103-118, Sevilla, ETS de Ingeniería Informática, 3-7 de Febrero, 2014,*, 2014.

[Clayer *et al.* 2013] Jean-Pierre Clayer, Claudine Toffolon, and Christophe Choquet. Patterns, pedagogical design schemes and process for instructional design. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*, pages 304–306. IEEE, 2013.

[Codd 1970] Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[Codd 1974] Edgar F Codd. *Seven steps to rendezvous with the casual user*. IBM Corporation, 1974.

[Coe 2007] Thomas S Coe. Using the Bloomberg professional system for finance classes. *Journal of Financial Education*, pages 48–62, 2007.

[Colby *et al.* 1971] Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. Artificial paranoia. *Artificial Intelligence*, 2(1):1–25, 1971.

[Colby 1975] Kenneth Mark Colby. *Artificial paranoia: a computer simulation of paranoid process*. Pergamon Press, 1975.

[Collin 2003] Peter Collin. *Dictionary of banking and finance*. Psychology Press, London, 2003.

[Collins and Halverson 2018] Allan Collins and Richard Halverson. *Rethinking education in the age of technology: The digital revolution and schooling in America*. Teachers College Press, 2018.

[Collobert and Weston 2008] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[Collobert *et al.* 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

[Connolly and Begg 2005] Thomas M Connolly and Carolyn E Begg. *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.

[Connolly and Begg 2006] Thomas M Connolly and Carolyn E Begg. A constructivist-based approach to teaching database analysis and design. *Journal of Information Systems Education*, 17(1):43, 2006.

[CoreNLP 2016] Stanford CoreNLP. a suite of core NLP tools. *URL http://nlp. stanford. edu/-software/corenlp. shtml (Last accessed: 2019-09-06)*, page 3, 2016.

[Cormen *et al.* 2009] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, USA, 3rd edition, 2009.

[Coronel and Morris 2016] Carlos Coronel and Steven Morris. *Database systems: design, implementation, & management*. Cengage Learning, 2016.

[Corritore and Wiedenbeck 1991] Cynthia L Corritore and Susan Wiedenbeck. What do novices learn during program comprehension? *International Journal of Human-Computer Interaction*, 3(2):199–222, 1991.

[Costa-Jussa and Fonollosa 2016] Marta R Costa-Jussa and José AR Fonollosa. Character-based neural machine translation. *arXiv preprint arXiv:1603.00810*, 2016.

[Couderc and Ferrero 2015] Benoît Couderc and Jérémy Ferrero. fr2SQL: Interrogation de bases de données en français. In *22ème Traitement Automatique des Langues Naturelles*, pages 1–13, 2015.

[Covington *et al.* 1994] Michael A Covington, Barbara J Grosz, and Fernando CN Pereira. *Natural language processing for Prolog programmers*. Prentice hall Englewood Cliffs (NJ), 1994.

[Crowe *et al.* 2008] Alison Crowe, Clarissa Dirks, and Mary Pat Wenderoth. Biology in Bloom: implementing Bloom's taxonomy to enhance student learning in biology. *CBE—Life Sciences Education*, 7(4):368–381, 2008.

[Cruse 2011] Alan Cruse. Meaning in language: An introduction to semantics and pragmatics. 2011.

[Dahl *et al.* 2012] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2012.

[Daigle 2007] B. Daigle. Enterprise to enterprise instant messaging. *Google Patents, US Patent App. 11/300,981*, 2007.

[Dale *et al.* 2000] Robert Dale, Hermann Moisl, and Harold Somers. *Handbook of natural language processing*. CRC Press, 2000.

[Dale 2019] Robert Dale. NLP commercialisation in the last 25 years. *Natural Language Engineering*, 25(3):419–426, 2019.

[Danaparamita and Gatterbauer 2011] Jonathan Danaparamita and Wolfgang Gatterbauer. QueryViz: helping users understand SQL queries and their patterns. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 558–561. ACM, 2011.

[Dandapat 2007] Sandipan Dandapat. Part of specch tagging and chunking with maximum entropy model. In *Proceedings of the IJCAI Workshop on Shallow Parsing for South Asian Languages*, pages 29–32, 2007.

[Dann *et al.* 2008] Wanda P Dann, Stephen Cooper, and Randy Pausch. *Learning to program with Alice*. Prentice Hall Press, 2008.

[Dann *et al.* 2011] Wanda P Dann, Stephen Cooper, and Randy Pausch. *Learning to Program with Alice (w/CD ROM)*. Prentice Hall Press, 2011.

[Davey 2017] Graham Davey. *Applications of conditioning theory*. Routledge, 2017.

[Davis *et al.* 1952] KH Davis, R Biddulph, and Stephen Balashek. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642, 1952.

[Davis 1997] Philip Davis. What computer skills do employees expect from recent college graduates? *THE Journal (Technological Horizons in Education)*, 25(2):74, 1997.

[De Boer and Badke-Schaub 2013] Robert J De Boer and Petra Badke-Schaub. Interaction between emotions and mental models in engineering and design activities. In *Emotional Engineering vol. 2*, pages 149–164. Springer, 2013.

[De Houwer *et al.* 2013] Jan De Houwer, Dermot Barnes-Holmes, and Agnes Moors. What is learning? on the nature and merits of a functional definition of learning. *Psychonomic bulletin & review*, 20(4):631–642, 2013.

[De Raadt *et al.* 2006] Michael De Raadt, Stijn Dekeyser, and Tien Yu Lee. Do students SQLify? improving learning outcomes with peer review and enhanced computer assisted assessment of querying skills. In *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*, pages 101–108. ACM, 2006.

[De Raadt *et al.* 2007] Michael De Raadt, Stijn Dekeyser, and Tien Yu Lee. A system employing peer review and enhanced computer assisted assessment of querying skills. *Informatics in education*, 6(1):163, 2007.

[de Silva 2017] NH Nisansa D de Silva. Relational databases and biomedical big data. *Bioinformatics in MicroRNA Research*, pages 69–81, 2017.

[Dearden and Finlay 2006] Andy Dearden and Janet Finlay. Pattern languages in HCI: A critical review. *Human–computer interaction*, 21(1):49–102, 2006.

[Dekeyser *et al.* 2007] Stijn Dekeyser, Michael de Raadt, and Tien Yu Lee. Computer assisted assessment of SQL query skills. In *Proceedings of the eighteenth conference on Australasian database-Volume 63*, pages 53–62. Australian Computer Society, Inc., 2007.

[Deng *et al.* 2013] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent advances in deep learning for speech research at microsoft. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8604–8608. IEEE, 2013.

[Deng *et al.* 2014] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[Di Persio and Honchar 2016] Luca Di Persio and Oleksandr Honchar. Artificial neural networks approach to the forecast of stock market price movements. *International Journal of Economics and Management Systems*, 1, 2016.

[Dietrich *et al.* 1997] Suzanne W Dietrich, Eric Eckert, and Kevin Piscator. WinRDBI: a windows-based relational database educational tool. In *ACM Special Interest Group on Computer Science Education Bulletin*, volume 29, pages 126–130. ACM, 1997.

[Digate *et al.* 2013] Charles J Digate, Christopher F Herot, Tonytip Ketudat, Alexis M Kopikis, and Daniel J Teven. System and method for immediate and delayed real-time communication activities using availability data from communication through an external instant messaging system. *US Patent 8,375,092*, 2013.

[Do *et al.* 2014] Quan Do, Rajeev K Agrawal, Dhana Rao, and Venkat N Gudivada. Automatic generation of SQL queries. *American Society for Engineering Education, Paper ID*, 8958, 2014.

[Dollinger 2010] Robert Dollinger. SQL lightweight tutoring module–semantic analysis of SQL queries based on XML representation and LINQ. In *EdMedia: World Conference on Educational Media and Technology*, pages 3323–3328. Association for the Advancement of Computing in Education, 2010.

[Dömösi *et al.* 2016] Pál Dömösi, Szilárd Zsolt Fazekas, and Masami Ito. On Chomsky Hierarchy of Palindromic Languages. *Acta Cybern.*, 22(3):703–713, 2016.

[Done and Murphy 2018] Elizabeth J Done and Mike Murphy. The responsibilisation of teachers: a neoliberal solution to the problem of inclusion. *Discourse: Studies in the Cultural Politics of Education*, 39(1):142–155, 2018.

[Dubey *et al.* 2016] Mohnish Dubey, Sourish Dasgupta, Ankit Sharma, Konrad Höffner, and Jens Lehmann. Asknow: A framework for natural language query formalization in SPARQL. In *European Semantic Web Conference*, pages 300–316. Springer, 2016.

[Dueck 2005] Gunter Dueck. *Dueck's Trilogie: Omnisophie – Supramanie – Topothesie*. Springer, Berlin, Germany, 2005. http://www.omnisophie.com.

[Duffy and Jonassen 2013] Thomas M Duffy and David H Jonassen. *Constructivism and the technology of instruction: A conversation*. Routledge, 2013.

[Duh 2018] Kevin Duh. *Bayesian Analysis in Natural Language Processing*, 2018.

[Dumais *et al.* 1998] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. 1998.

[Duncan 2015] Andrew Duncan. Demonstration of Bloomberg terminal and instant Bloomberg messaging. *Personal Communication*, 2015.

[Dunn and Dunn 1978] Rita Stafford Dunn and Kenneth J Dunn. *Teaching students through their individual learning styles: A practical approach*. Prentice Hall, 1978.

[Eberl and Kaiser 2018] Maximilian Eberl and Stephan Kaiser. *Organizational Routines Meet Experimental Psycholo: The Role of Implicit Learning in the Modification of Organizational Routines*. Springer, 2018.

[Eden *et al.* 2018] Amnon H Eden, Epameinondas Gasparis, Jon Nicholson, and Rick Kazman. Round-trip engineering with the two-tier programming toolkit. *Software Quality Journal*, 26(2):249–271, 2018.

[El-Affendi 2018] Mohammed A El-Affendi. The generative power of Arabic morphology and implications: A case for pattern orientation in Arabic corpus annotation and a proposed pattern ontology. In *5th International Symposium on Data Mining Applications*, pages 36–45. Springer, 2018.

[Elder 2009] Marvin Elder. *Natural language database querying*, April 30 2009. US Patent App. 12/004,255.

[Elhemali *et al.* 2007] Mostafa Elhemali, César A Galindo-Legaria, Torsten Grabs, and Milind M Joshi. Execution strategies for SQL subqueries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 993–1004. ACM, 2007.

[Ellis and Dix 2007] Geoffrey Ellis and Alan Dix. A taxonomy of clutter reduction for information visualisation. *IEEE transactions on visualization and computer graphics*, 13(6):1216–1223, 2007.

[Ellis and Mansmann 2010] Geoffrey Ellis and Florian Mansmann. Mastering the information age solving problems with visual analytics. In *Eurographics*, volume 2, page 5, 2010.

[Elmasri 2017] Ramez Elmasri. *Fundamentals of database systems*. 2017.

[ElSayed 2015] Khaled Nasser ElSayed. An Arabic natural language interface system for a database of the Holy Quran. *International Journal of Advanced Research in Artificial Intelligence*, 4(7):9–14, 2015.

[Elsom-Cook 1987] Mark Elsom-Cook. Intelligent computer-aided instruction research at the open university. cite report no. 10. 1987.

[Engeström and Sannino 2012] Yrjö Engeström and Annalisa Sannino. Whatever happened to process theories of learning? *Learning, Culture and Social Interaction*, 1(1):45–56, 2012.

[Entwistle and Ramsden 2015] Noel Entwistle and Paul Ramsden. *Understanding student learning (Routledge revivals)*. Routledge, 2015.

[Eriksson *et al.* 2005] Kent Eriksson, Katri Kerem, and Daniel Nilsson. Customer acceptance of Internet banking in Estonia. *International Journal of Bank Marketing*, 23(2):200–216, 2005.

[Ertmer and Newby 2013] Peggy A Ertmer and Timothy J Newby. Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance improvement quarterly*, 26(2):43–71, 2013.

[Etzioni and Kautz 2002] Oren Etzioni and Henry Kautz. High precision natural language interfaces to databases: a graph theoretic approach draft. 2002.

[Evgeniou and Pontil 2001] Theodoros Evgeniou and Massimiliano Pontil. Workshop on support vector machines: theory and applications. 2001.

[Explosion 2017] AI Explosion. spaCy-industrial-strength natural language processing in Python. *URL: https://spacy. io*, 2017.

[Ezpeleta *et al.* 2016] Enaitz Ezpeleta, Urko Zurutuza, and José María Gómez Hidalgo. Does sentiment analysis help in Bayesian spam filtering? In *International Conference on Hybrid Artificial Intelligence Systems*, pages 79–90. Springer, 2016.

[Fagin *et al.* 2015] Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Document spanners: A formal approach to information extraction. *Journal of the ACM (JACM)*, 62(2):12, 2015.

[Fang *et al.* 2005] Xiang Fang, Sooun Lee, and Seokha Koh. Transition of knowledge/skills requirement for entry-level is professionals: An exploratory study based on recruiters' perception. *Journal of Computer Information Systems*, 46(1):58–70, 2005.

[Farajian *et al.* 2017] M Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137, 2017.

[Faroult and Robson 2006] Stephane Faroult and Peter Robson. *The art of SQL*. " O'Reilly Media, Inc.", 2006.

[Felder *et al.* 1988] Richard M Felder, Linda K Silverman, et al. Learning and teaching styles in engineering education. *Engineering education*, 78(7):674–681, 1988.

[Felicia and Pitt 2009] Patrick Felicia and IJ Pitt. Profiling users in educational games. *Games-based learning advancement for multisensory human computer interfaces: Techniques and effective practices, Idea-Group, Hershey*, 2009.

[Feng *et al.* 2017] Huan Feng, Kassem Fawaz, and Kang G Shin. Continuous authentication for voice assistants. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 343–355. ACM, 2017.

[Ferguson 2014] Mike Ferguson. E paper. *BMJ*, pages 72–73, 2014.

[Fernau *et al.* 2015] Henning Fernau, Meenakshi Paramasivan, and Markus L Schmid. Jumping finite automata: characterizations and complexity. In *International Conference on Implementation and Application of Automata*, pages 89–101. Springer, 2015.

[Fincher 1999] Sally Fincher. What are we doing when we teach programming? In *FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No. 99CH37011*, volume 1, pages 12A4–1. IEEE, 1999.

[Fleming *et al.* 2011] Sandra Fleming, Gabrielle Mckee, and Sylvia Huntley-Moore. Undergraduate nursing students' learning styles: A longitudinal study. *Nurse education today*, 31(5):444–449, 2011.

[Folland 2016] Kristin Annabel Torjussen Folland. *viSQLizer: An interactive visualizer for learning SQL*. Master's thesis, 2016.

[Forehand 2010] Mary Forehand. Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology*, 41(4):47–56, 2010.

[Foster and Godbole 2016] Elvis C Foster and Shripad Godbole. Overview of SQL. In *Database Systems*, pages 205–209. Springer, 2016.

[Fraser 2015] Neil Fraser. Ten things we've learned from Blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, pages 49–50. IEEE, 2015.

[Friedl 2006] Jeffrey EF Friedl. *Mastering regular expressions*. "O'Reilly Media, Inc.", 2006.

[Fu *et al.* 2016] Rongrong Fu, Hong Wang, and Wenbo Zhao. Dynamic driver fatigue detection using hidden Markov model in real driving condition. *Expert Systems with Applications*, 63:397–411, 2016.

[Furst *et al.* 2002] Karen Furst, William W Lang, and Daniel E Nolle. Internet banking. *Journal of Financial Services Research*, 22(1-2):95–117, 2002.

[Gales *et al.* 2008] Mark Gales, Steve Young, et al. The application of hidden Markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304, 2008.

[Gantayat *et al.* 2019] Neelamadhav Gantayat, Diptikalyan Saha, Jaydeep Sen, and Senthil Mani. Goal-based ontology creation for natural language querying in sap-erp platform. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 231–237. ACM, 2019.

[Ganty and Valero 2019] Pierre Ganty and Pedro Valero. Regular expression search on compressed text. In *2019 Data Compression Conference (DCC)*, pages 528–537. IEEE, 2019.

[Gao *et al.* 2018] Chong-zhi Gao, Qiong Cheng, Pei He, Willy Susilo, and Jin Li. Privacy-preserving naive Bayes classifiers secure against the substitution-then-comparison attack. *Information Sciences*, 444:72–88, 2018.

[Garcia-Molina and Salem 1992] Hector Garcia-Molina and Kenneth Salem. Main memory database systems: An overview. *IEEE Transactions on Knowledge & Data Engineering*, (6):509–516, 1992.

[Gardner *et al.* 2018] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018.

[Garner and Mariani 2015] Philip Garner and John Amedeo Mariani. Learning SQL in steps. *Journal on Systemics, Cybernetics and Informatics*, 13(4):19–24, 2015.

[Garner 2003] Stuart Garner. Learning resources and tools to aid novices learn programming. In *Informing science & information technology education joint conference (INSITE)*, pages 213–222, 2003.

[Genise *et al.* 2019] Nicholas Genise, Craig Gentry, Shai Halevi, Baiyu Li, and Daniele Micciancio. Homomorphic encryption for finite automata. *IACR Cryptology ePrint Archive*, 2019:176, 2019.

[Gentner and Stevens 2014] Dedre Gentner and Albert L Stevens. *Mental models*. Psychology Press, 2014.

[Ghai *et al.* 2019] Ridhima Ghai, Sakshum Kumar, and Avinash Chandra Pandey. Spam detection using rating and review processing method. In *Smart Innovations in Communication and Computational Sciences*, pages 189–198. Springer, 2019.

[Gil and Lenz 2007] Joseph Yossi Gil and Keren Lenz. Simple and safe SQL queries with C++ templates. In *Proceedings of the 6th International Conference on Generative Programming and Component Engineering*, pages 13–24. ACM, 2007.

[Gill 2019] Navdeep Singh Gill. *Overview of artificial intelligence and natural language processing*, 2019.

[Giordani 2008] Alessandra Giordani. Mapping natural language into SQL in a NLIDB. In *International Conference on Application of Natural Language to Information Systems*, pages 367–371. Springer, 2008.

[Glas *et al.* 2012] D Glas, Satoru Satake, Takayuki Kanda, and Norihiro Hagita. An interaction design framework for social robots. In *Robotics: Science and Systems*, volume 7, page 89, 2012.

[Godinez and Jamil 2018] Josue Espinosa Godinez and Hasan M Jamil. Meet Cyrus-the Query by Voice mobile assistant for the tutoring and formative assessment of SQL learners. *arXiv preprint arXiv:1811.04160*, pages 1–6, 2018.

[Goel *et al.* 2019] Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *arXiv preprint arXiv:1907.00883*, 2019.

[Gogte *et al.* 2016] Vaibhav Gogte, Aasheesh Kolli, Michael J Cafarella, Loris D'Antoni, and Thomas F Wenisch. Hare: Hardware accelerator for regular expressions. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, page 44. IEEE Press, 2016.

[Goldberg and Elhadad 2008] Yoav Goldberg and Michael Elhadad. splitSVM: Fast, space-efficient, non-heuristic, polynomial kernel computation for NLP applications. In *Proceedings of ACL-08: HLT, Short Papers*, pages 237–240, 2008.

[Gooch 2012] Phil Gooch. BADREX: In situ expansion and coreference of biomedical abbreviations using dynamic regular expressions. *arXiv preprint arXiv:1206.4522*, 2012.

[Goodfellow *et al.* 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[Gorman 2002] Michael E Gorman. Types of knowledge and their roles in technology transfer. *The Journal of Technology Transfer*, 27(3):219–231, 2002.

[Goyal *et al.* 2018] Palash Goyal, Sumit Pandey, and Karan Jain. Deep learning for natural language processing. *Apress, Berkeley*, 2018.

[Goyvaerts 2006] Jan Goyvaerts. *Regular Expressions: The Complete Tutorial*. Lulu Press, 2006.

[Grad 2012] Burton Grad. Relational database management systems: The formative years [guest editor's introduction]. *IEEE Annals of the History of Computing*, 34(4):7–8, 2012.

[Graesser *et al.* 2012] Arthur C Graesser, Mark W Conley, and Andrew Olney. Intelligent tutoring systems. 2012.

[Gray and Malins 2016] Carole Gray and Julian Malins. *Visualizing research: A guide to the research process in art and design*. Routledge, 2016.

[Gredler 1992] Margaret E Gredler. *Learning and instruction: Theory into practice*. Macmillan New York, 1992.

[Green Jr *et al.* 1961] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

[Grefenstette *et al.* 2000] Gregory Grefenstette, Anne Schiller, and Salah Aït-Mokhtar. Recognizing lexical patterns in text. In *Lexicon Development for Speech and Language Processing*, pages 141–168. Springer, 2000.

[Grillenberger and Brinda 2012] Andreas Grillenberger and Torsten Brinda. eledSQL: a new web-based learning environment for teaching databases and SQL at secondary school level. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, pages 101–104. ACM, 2012.

[Groff *et al.* 2002] James R Groff, Paul N Weinberg, et al. *SQL: the complete reference*, volume 2. McGraw-Hill/Osborne, 2002.

[Große and Renkl 2007] Cornelia S Große and Alexander Renkl. Finding and fixing errors in worked examples: Can this foster learning outcomes? *Learning and Instruction*, 17(6):612–634, 2007.

[Grover *et al.* 2015] Shuchi Grover, Roy Pea, and Stephen Cooper. Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2):199–237, 2015.

[Grune *et al.* 2012] Dick Grune, Kees Van Reeuwijk, Henri E Bal, Ceriel JH Jacobs, and Koen Langendoen. *Modern compiler design*. Springer Science & Business Media, 2012.

[Grust and Scholl 1999] Torsten Grust and Marc H Scholl. How to comprehend queries functionally. *Journal of Intelligent Information Systems*, 12(2-3):191–218, 1999.

[Guimaraes 2006] Mario Guimaraes. The kennesaw database courseware (kdc): strong points, weak points, and experience using it in a classroom environment. *Journal of Computing Sciences in Colleges*, 21(3):91–96, 2006.

[Gulwani *et al.* 2015] Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H Muggleton, Ute Schmid, and Benjamin Zorn. Inductive programming meets the real world. *Communications of the ACM*, 58(11):90–99, 2015.

[Gussenhoven and Jacobs 2017] Carlos Gussenhoven and Haike Jacobs. *Understanding phonology*. Routledge, 2017.

[Haghighi *et al.* 2018] Sepand Haghighi, Masoomeh Jasemi, Shaahin Hessabi, and Alireza Zolanvari. Pycm: Multiclass confusion matrix library in Python. *J. Open Source Software*, 3(25):729, 2018.

[Haiduc *et al.* 2010] S. Haiduc, J. Aponte, and A. Marcus. Supporting program comprehension with source code summarization. *ACM/IEEE 32nd International Conference on Software Engineering*, 2:223–226, 2010.

[Hale 2001] John Hale. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.

[Hamilton *et al.* 2018] James Hamilton, Manuel Gonzalez Berges, Jean-Charles Tournier, and Brad Schofield. Scada statistics monitoring using the elastic stack (elasticsearch, logstash, kibana). 2018.

[Han *et al.* 2013] Bo Han, Paul Cook, and Timothy Baldwin. unimelb: Spanish text normalisation. In *Tweet Normalisation Workshop at Sociedad Espanola para el Procesamiento del Lenguaje Natural (SEPLN)*, pages 32–36, 2013.

[Hansen and Reich 2015] John D Hansen and Justin Reich. Democratizing education? examining access and usage patterns in massive open online courses. *Science*, 350(6265):1245–1248, 2015.

[Hanus and Krone 2017] Michael Hanus and Julia Krone. A typeful integration of SQL into curry. *arXiv preprint arXiv:1701.00631*, 2017.

[Harasim 2017] Linda Harasim. *Learning theory and online technologies*. Routledge, 2017.

[Harden 2017] Cynthia Harden. Regular expressions for IT men. 2017.

[Harrison 2015] Guy Harrison. Languages and programming interfaces. In *Next Generation Databases*, pages 167–190. Springer, 2015.

[Hashem *et al.* 2015] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, 2015.

[Hatami 2017] Zahra Hatami. Towards an understanding of the mental model process while writing SQL queries. 2017.

[Hayati *et al.* 2010] Pedram Hayati, Vidyasagar Potdar, Alex Talevski, Nazanin Firoozeh, Saeed Sarenche, and Elham A Yeganeh. Definition of spam 2.0: New spamming boom. In *4th IEEE International Conference on Digital Ecosystems and Technologies*, pages 580–584. IEEE, 2010.

[Hein 1999] George E Hein. The constructivist museum. *The educational role of the museum*, 2:73–79, 1999.

[Heinroth and Minker 2012] Tobias Heinroth and Wolfgang Minker. *Introducing spoken dialogue systems into Intelligent Environments*. Springer Science & Business Media, 2012.

[Heller 2019a] Jon Heller. Modify data with advanced DML. In *Pro Oracle SQL Development*, pages 191–218. Springer, 2019.

[Heller 2019b] Jon Heller. Understand relational databases. In *Pro Oracle SQL Development*, pages 3–28. Springer, 2019.

[Heller 2019c] Jon Heller. Write large SQL statements. In *Pro Oracle SQL Development*, pages 309–325. Springer, 2019.

[Hewitt and Manning 2019] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019.

[Hinton *et al.* 2012] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[Hirschberg and Manning 2015] Julia Hirschberg and Christopher D Manning. Advances in natural language processing. *Science*, 349(6245):261–266, 2015.

[Hirschman and Gaizauskas 2001] Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300, 2001.

[Hirtle 1996] Jeannine St Pierre Hirtle. Social constructivism. *English Journal*, 85(1):91, 1996.

[Hmelo-Silver 2004] Cindy E Hmelo-Silver. Problem-based learning: What and how do students learn? *Educational psychology review*, 16(3):235–266, 2004.

[Hoang *et al.* 2015] Duc Tam Hoang, Minh Le Nguyen, and Son Bao Pham. L2S: Transforming natural language questions into SQL queries. In *2015 Seventh International Conference on Knowledge and Systems Engineering*, pages 85–90. IEEE, 2015.

[Hogan 2018] Rex Hogan. *A practical guide to database design*. Chapman and Hall/CRC, 2018.

[Hoic-Bozic *et al.* 2009] Natasa Hoic-Bozic, Vedran Mornar, and Ivica Boticki. A blended learning approach to course design and implementation. *IEEE Transactions on Education*, 52(1):19–30, 2009.

[Honey *et al.* 1992] Peter Honey, Alan Mumford, et al. The manual of learning styles. 1992.

[Horn *et al.* 2017] Ilana Seidel Horn, Brette Garner, Britnie Delinger Kane, and Jason Brasel. A taxonomy of instructional learning opportunities in teachers' workgroup conversations. *Journal of Teacher Education*, 68(1):41–54, 2017.

[Hosu *et al.* 2018] Ionel Alexandru Hosu, Radu Cristian Alexandru Iacob, Florin Brad, Stefan Ruseti, and Traian Rebedea. Natural language interface for databases using a Dual-Encoder model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 514–524, 2018.

[Hoy 2018] Matthew B Hoy. Alexa, Siri, Cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

[Hu and Shepherd 2013] Helen H Hu and Tricia D Shepherd. Using POGIL to help students learn to program. *ACM Transactions on Computing Education (TOCE)*, 13(3):13, 2013.

[Hu *et al.* 2007] Ying Hu, Seema Sundara, and Jagannathan Srinivasan. Supporting time-constrained SQL queries in Oracle. In *Proceedings of the 33rd International Conference on very Large Databases*, pages 1207–1218. VLDB Endowment, 2007.

[Huang *et al.* 2014a] Chaochao Huang, Xipeng Qiu, and Xuanjing Huang. Text classification with document embeddings. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 131–140. Springer, 2014.

[Huang *et al.* 2014b] James Huang, Stephanie Rogers, and Eunkwang Joo. Improving restaurants by extracting subtopics from Yelp reviews. *iConference 2014 (Social Media Expo)*, 2014.

[Huang *et al.* 2014c] Xuedong Huang, James Baker, and Raj Reddy. A historical perspective of speech recognition. *Communications of the ACM*, 57(1):94–103, 2014.

[Huang *et al.* 2015] Ting-Hao Kenneth Huang, Walter S Lasecki, and Jeffrey P Bigham. Guardian: A crowd-powered spoken dialog system for web apis. In *Third AAAI conference on human computation and crowdsourcing*, 2015.

[Hughes *et al.* 2017] Mark Hughes, I Li, Spyros Kotoulas, and Toyotaro Suzumura. Medical text classification using convolutional neural networks. *Stud Health Technol Inform*, 235:246–250, 2017.

[Hung 2001] David Hung. Theories of learning and computer-mediated instructional technologies. *Educational Media International*, 38(4):281–287, 2001.

[Hurst *et al.* 2013] Beth Hurst, Randall Wallace, and Sarah B Nixon. The impact of social interaction on student learning. *Reading Horizons*, 52(4):5, 2013.

[Hutchins and Somers 1992] William John Hutchins and Harold L Somers. *An introduction to machine translation*, volume 362. Academic Press London, 1992.

[Hutchins 1986] William John Hutchins. *Machine translation: past, present, future*. Ellis Horwood Chichester, 1986.

[Idris *et al.* 2015] Ismaila Idris, Ali Selamat, Ngoc Thanh Nguyen, Sigeru Omatu, Ondrej Krejcar, Kamil Kuca, and Marek Penhaker. A combined negative selection algorithm–particle swarm optimization for an email spam detection system. *Engineering Applications of Artificial Intelligence*, 39:33–44, 2015.

[Iedemska *et al.* 2014] Jane Iedemska, Gianluca Stringhini, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. The tricks of the trade: What makes spam campaigns successful? In *2014 IEEE Security and Privacy Workshops*, pages 77–83. IEEE, 2014.

[Inge 2013] Andre Inge. Hidden Markov models. 2013.

[Ireland 2015] Katherine Ireland. The Bloomberg terminal: Mission critical or a "very expensive facebook"? *Harvard Business School Open Forum, Harvard Business School*, 2015.

[Ismail and Homsi 2018] Walaa Saber Ismail and Masun Nabhan Homsi. DAWQAS: A dataset for Arabic why question answering system. *Procedia computer science*, 142:123–131, 2018.

[Jackson and Moulinier 2007] Peter Jackson and Isabelle Moulinier. *Natural language processing for online applications*. John Benjamins, 2007.

[Jacobs 2014] Paul S Jacobs. *Text-based intelligent systems: Current research and practice in information extraction and retrieval*. Psychology Press, 2014.

[Jäger and Rogers 2012] Gerhard Jäger and James Rogers. Formal language theory: refining the chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970, 2012.

[James 2006] Mary James. Assessment, teaching and theories of learning. *Assessment and learning*, 47:60, 2006.

[Jamil 2017] Hasan M Jamil. Knowledge rich natural language queries over structured biological databases. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 352–361. ACM, 2017.

[Jangid *et al.* 2018] Hitkul Jangid, Shivangi Singhal, Rajiv Ratn Shah, and Roger Zimmermann. Aspect-based financial sentiment analysis using deep learning. In *Companion Proceedings of the The Web Conference 2018*, pages 1961–1966. International World Wide Web Conferences Steering Committee, 2018.

[Jansen *et al.* 2009] Bernard J Jansen, Danielle Booth, and Brian Smith. Using the taxonomy of cognitive learning to model online searching. *Information Processing & Management*, 45(6):643–663, 2009.

[Javed 2009] Faizan Javed. *Techniques for Context-Free Grammar Induction and Applications: Application of novel inference algorithms to software maintenance problems*. VDM Verlag, 2009.

[Jenkins 2002] Tony Jenkins. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58. Citeseer, 2002.

[Johnson and Fuller 2006] Colin G Johnson and Ursula Fuller. Is Bloom's taxonomy appropriate for computer science? In *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*, pages 120–123. ACM, 2006.

[Johnson and Soloway 1985] W Lewis Johnson and Elliot Soloway. PROUST: Knowledge-based program understanding. *IEEE Transactions on Software Engineering*, (3):267–275, 1985.

[Johnson-Laird 2010] Philip N Johnson-Laird. Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43):18243–18250, 2010.

[Jones *et al.* 2016] Kiku Jones, Lori NK Leonard, and Guido Lang. Desired skills for entry level IS positions: Identification and assessment. *Journal of Computer Information Systems*, pages 1–7, 2016.

[Jost *et al.* 2005] Timothée Jost, Nabil Ouerhani, Roman Von Wartburg, René Müri, and Heinz Hügli. Assessing the contribution of color in visual attention. *Computer Vision and Image Understanding*, 100(1-2):107–123, 2005.

[Juang and Rabiner 1991] Biing Hwang Juang and Laurence R Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[Julavanich *et al.* 2019] Thanakrit Julavanich, Srinual Nalintippayawong, and Kanokwan Atchariyachanvanich. RSQLg: The reverse SQL question generation algorithm. In *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 908–912. IEEE, 2019.

[Jungbluth *et al.* 2018] Jan Jungbluth, Rolf Krieger, Wolfgang Gerke, and Peter Plapper. Combining virtual and robot assistants - a case study about integrating Amazon's Alexa as a voice interface in robotics. In *Robotix-Academy Conference for Industrial Robotics*, page 5. Shaker, 2018.

[Jurafsky and Manning 2012] Dan Jurafsky and Christopher Manning. Natural language processing. *Instructor*, 212(998):3482, 2012.

[Jurafsky 2000] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[Jusoh and Alfawareh 2012] Shaidah Jusoh and Hejab M Alfawareh. Techniques, applications and challenging issue in text mining. *International Journal of Computer Science Issues (IJCSI)*, 9(6):431, 2012.

[Kalina and Powell 2009] Cody Kalina and KC Powell. Cognitive and social constructivism: Developing tools for an effective classroom. *Education*, 130(2):241–250, 2009.

[Kantorowitz 2014] Eliezer Kantorowitz. Verbal use case specifications for informal requirements elicitation. In *Building Bridges: HCI, Visualization, and Non-formal Modeling*, pages 165–174. Springer, 2014.

[Kapetanios 2008] Epaminondas Kapetanios. *Natural Language and Information Systems: 13th International Conference on Applications of Natural Language to Information Systems, NLDB 2008 London, UK, June 24-27, 2008, Proceedings*, volume 5039. Springer Science & Business Media, 2008.

[Kapur *et al.* 2018] Arnav Kapur, Shreyas Kapur, and Pattie Maes. Alterego: A personalized wearable silent speech interface. In *23rd International Conference on Intelligent User Interfaces*, pages 43–53. ACM, 2018.

[Karhumäki 2007] Juhani Karhumäki. *Automata and formal languages*, 2007.

[Kari 2013] Jarkko Kari. Automata and formal languages. *Fall semester*, 2013.

[Katz and Shmallo 2016] Adi Katz and Ronit Shmallo. Learning from errors as a pedagogic approach for reaching a higher conceptual level in database modeling. In *International Conference on Advanced Information Systems Engineering*, pages 93–102. Springer, 2016.

[Kawash 2014] Jalal Kawash. Formulating second-order logic conditions in SQL. In *Proceedings of the 15th Annual Conference on Information technology education*, pages 115–120. ACM, 2014.

[Kearns *et al.* 1997] R Kearns, Stephen Shead, and Alan Fekete. A teaching system for SQL. In *Proceedings of the 2nd Australasian conference on Computer science education*, pages 224–231. ACM, 1997.

[Keim *et al.* 2008] Daniel A Keim, Florian Mansmann, Jörn Schneidewind, Jim Thomas, and Hartmut Ziegler. Visual analytics: Scope and challenges. In *Visual data mining*, pages 76–90. Springer, 2008.

[Kejriwal *et al.* 2017] Mayank Kejriwal, Jiayuan Ding, Runqi Shao, Anoop Kumar, and Pedro Szekely. Flagit: A system for minimally supervised human trafficking indicator mining. *arXiv preprint arXiv:1712.03086*, 2017.

[Kellems *et al.* 2016] Ryan O Kellems, Terisa P Gabrielsen, and Caroline Williams. Using visual organizers and technology: Supporting executive function, abstract language comprehension, and social learning. In *Technology and the Treatment of Children with Autism Spectrum Disorder*, pages 75–86. Springer, 2016.

[Kendall and Gal 2017] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[Kennan *et al.* 2005] Mary Anne Kennan, Fletcher Cole, Patricia Willard, and C.S Wilson. Changes in the workplace: transformation of the information professional? In *2nd Research Applications in Information and Library Studies Seminar, Canberra*, pages 1–19, 2005.

[Keogh 2006] Eamonn Keogh. Naive Bayes classifier. 2006.

[Këpuska and Bohouta 2017] Veton Këpuska and Gamal Bohouta. Comparing speech recognition systems (microsoft api, google api and cmu sphinx). *Int. J. Eng. Res. Appl*, 7(03):20–24, 2017.

[Keselj 2009] Vlado Keselj. *Speech and Language Processing Pearson Prentice Hall*, 2009.

[Khosravi *et al.* 2018] Pegah Khosravi, Ehsan Kazemi, Marcin Imielinski, Olivier Elemento, and Iman Hajirasouliha. Deep convolutional neural networks enable discrimination of heterogeneous digital pathology images. *EBioMedicine*, 27:317–328, 2018.

[Khurana *et al.* 2017] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *arXiv preprint arXiv:1708.05148*, 2017.

[Khuziakhmetov and Porchesku 2016] Anvar N Khuziakhmetov and Galina V Porchesku. Teaching listening comprehension: Bottom-up approach. *International journal of environmental and science education*, 11(8):1989–2001, 2016.

[Kinchin 2011] Ian M Kinchin. Visualising knowledge structures in biology: Discipline, curriculum and student understanding. *Journal of Biological Education*, 45(4):183–189, 2011.

[Kirillov 2017] Danil Kirillov. *Clustering Spam Emails Using Hadoop and FP-Trees*. PhD thesis, Carleton University, 2017.

[Kleerekoper and Schofield 2018] Anthony Kleerekoper and Andrew Schofield. SQL tester: an online SQL assessment tool and its impact. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 87–92. ACM, 2018.

[Klein *et al.* 2017] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

[Klein 2006a] Ewan Klein. Computational semantics in the natural language toolkit. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 26–33, 2006.

[Klein 2006b] Richard Klein. *The Wits Intelligent Teaching System (WITS): A smart lecture theatre to assess audience engagement*. PhD thesis, 2006.

[Kleiner *et al.* 2013] Carsten Kleiner, Christopher Tebbe, and Felix Heine. Automated grading and tutoring of SQL statements to improve student learning. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, pages 161–168. ACM, 2013.

[Knuth 1974] Donald E. Knuth. Computer Programming as an Art. *Communications of the ACM*, 17(12):667–673, 1974.

[Knuth 1976] Donald E. Knuth. Big Omicron and Big Omega and Big Theta. *SIGACT News*, 8(2):18–24, 1976.

[Koehn and Knowles 2017] Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*, 2017.

[Koehn *et al.* 2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180, 2007.

[Kokkalis *et al.* 2012] Andreas Kokkalis, Panagiotis Vagenas, Alexandros Zervakis, Alkis Simitsis, Georgia Koutrika, and Yannis Ioannidis. Logos: a system for translating queries into narratives. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 673–676. ACM, 2012.

[Kornacker *et al.* 2015] Marcel Kornacker, Alexander Behm, Victor Bittorf, Taras Bobrovytsky, Casey Ching, Alan Choi, Justin Erickson, Martin Grund, Daniel Hecht, Matthew Jacobs, et al. Impala: A modern, open-source SQL engine for Hadoop. In *7th Biennial Conference on Innovative Data Systems Research*, volume 1, page 9, 2015.

[Kotov 2017] Mykhailo Kotov. NLP resources for a rare language morphological analyzer: Danish case. In *Computational linguistics and intelligent systems (COLINS 2017)*. National Technical University «KhPI», 2017.

[Kotzé *et al.* 2008] Paula Kotzé, Karen Renaud, and Judy van Biljon. Don't do this–pitfalls in using anti-patterns in teaching human–computer interaction principles. *Computers & Education*, 50(3):979–1008, 2008.

[Kouemou and Dymarski 2011] Guy Leonard Kouemou and Dr Przemyslaw Dymarski. History and theoretical basics of hidden Markov models. *Hidden Markov Models, Theory and Applications*, 1, 2011.

[Krathwohl and Anderson 2001] David R Krathwohl and Lorin W Anderson. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman, 2001.

[Krathwohl 2002] David R Krathwohl. A revision of Bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218, 2002.

[Kreie and Ernst 2013] Jennifer Kreie and Bruce A Ernst. From database concepts to application: Use problem-based learning and Oracle development tools to facilitate learning. In *Proceedings of the Information Systems Educators Conference*, volume 2167, page 1435. Citeseer, 2013.

[Křivka and Meduna 2015] Zbyněk Křivka and Alexander Meduna. Jumping grammars. *International Journal of Foundations of Computer Science*, 26(06):709–731, 2015.

[Krusche *et al.* 2018] Stephan Krusche, Bruce Scharlau, Åsa Cajander, and Janet Hughes. 50 years of software engineering: challenges, results, and opportunities in its education. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 362–363. ACM, 2018.

[Kugler 2019] Logan Kugler. Being recognized everywhere. *Communications of the ACM*, 62(2):17–19, 2019.

[Kumar and Kanal 1988] Vipin Kumar and Laveen N Kanal. The CDP: A unifying formulation for heuristic search, dynamic programming, and branch-and-bound. In *Search in Artificial Intelligence*, pages 1–27. Springer, 1988.

[Kumaresan and Palanisamy 2017] T Kumaresan and C Palanisamy. E-mail spam classification using s-cuckoo search and support vector machine. *International Journal of Bio-Inspired Computation*, 9(3):142–156, 2017.

[Kupferman 2018] Orna Kupferman. Automata theory and model checking. In *Handbook of Model Checking*, pages 107–151. Springer, 2018.

[Kwon *et al.* 2019] Youngsung Kwon, Alexis Kwasinski, and Andres Kwasinski. Solar irradiance forecast using naïve Bayes classifier based on publicly available weather forecasting variables. *Energies*, 12(8):1529, 2019.

[Kyfonidis *et al.* 2017] Charalampos Kyfonidis, Nektarios Moumoutzis, and Stavros Christodoulakis. Block-C: A block-based programming teaching tool to facilitate introductory C programming courses. In *Global Engineering Education Conference (EDUCON), 2017 IEEE*, pages 570–579. IEEE, 2017.

[Lachman 1997] Sheldon J Lachman. Learning is a process: Toward an improved definition of learning. *The Journal of psychology*, 131(5):477–480, 1997.

[Lahtinen *et al.* 2005] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. A study of the difficulties of novice programmers. In *ACM SIGSCE Bulletin*, volume 37, pages 14–18. ACM, 2005.

[Lahtinen 2007] Essi Lahtinen. A categorization of novice programmers: A cluster analysis study. In *PPIG*, volume 16, pages 32–41, 2007.

[Lai *et al.* 2008] Jennifer Lai, Clare-Marie Karat, and Nicole Yankelovich. Conversational speech interfaces and technologies. *The Human-Computer Interaction Handbook,*, pages 381–391, 2008.

[Lane 2012] H Chad Lane. Cognitive models of learning. *Encyclopedia of the sciences of learning*, pages 608–610, 2012.

[Laurillard 2013] Diana Laurillard. *Teaching as a design science: Building pedagogical patterns for learning and technology*. Routledge, 2013.

[Lavbič *et al.* 2017] Dejan Lavbič, Tadej Matek, and Aljaž Zrnec. Recommender system for learning SQL using hints. *Interactive Learning Environments*, 25(8):1048–1064, 2017.

[Lawrence 2014] Ramon Lawrence. Integration and virtualization of relational SQL and NoSQL systems including mySQL and MongoDB. In *2014 International Conference on Computational Science and Computational Intelligence*, volume 1, pages 285–290. IEEE, 2014.

[LeCun *et al.* 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[Lee *et al.* 2013] Yun Young Lee, Nicholas Chen, and Ralph E Johnson. Drag-and-drop refactoring: intuitive and efficient program transformation. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 23–32. IEEE Press, 2013.

[Leech 1992] Geoffrey Neil Leech. 100 million words of English: the British National Corpus (BNC). 1992.

[Lehnert and Ringle 2014] Wendy G Lehnert and Martin H Ringle. *Strategies for natural language processing*. Psychology Press, 2014.

[Lemut *et al.* 2013] Enrica Lemut, Benedict DuBoulay, and Giuliana Dettori. *Cognitive models and intelligent environments for learning programming*, volume 111. Springer Science & Business Media, 2013.

[Letovsky 1987] Stanley Letovsky. Cognitive processes in program comprehension. *Journal of Systems and software*, 7(4):325–339, 1987.

[Levene and Loizou 2012] Mark Levene and George Loizou. *A guided tour of relational databases and beyond*. Springer Science & Business Media, 2012.

[Levulis *et al.* 2018] Samuel J Levulis, Patricia R DeLucia, and So Young Kim. Effects of touch, voice, and multimodal input, and task load on multiple - UAV monitoring performance during simulated manned-unmanned teaming in a military helicopter. *Human factors*, 60(8):1117–1129, 2018.

[Li and Jagadish 2014a] Fei Li and Hosagrahar V Jagadish. NaLIR: an interactive natural language interface for querying relational databases. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 709–712. ACM, 2014.

[Li and Jagadish 2014b] Fei Li and HV Jagadish. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84, 2014.

[Li and Jagadish 2016] Fei Li and HV Jagadish. Understanding natural language queries over relational databases. *ACM Special Interest Group on Management of Data Record*, 45(1):6–13, 2016.

[Li *et al.* 2008] Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and HV Jagadish. Regular expression learning for information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 21–30. Association for Computational Linguistics, 2008.

[Li *et al.* 2010] Linlin Li, Benjamin Roth, and Caroline Sporleder. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1138–1147. Association for Computational Linguistics, 2010.

[Li *et al.* 2016] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[Li *et al.* 2018] Tong Li, Jin Li, Zheli Liu, Ping Li, and Chunfu Jia. Differentially private naive Bayes learning over multiple data sources. *Information Sciences*, 444:89–104, 2018.

[Li *et al.* 2019] Jingjing Li, Wenlu Wang, Wei-Shinn Ku, Yingtao Tian, and Haixun Wang. SpatialNLI: A spatial domain natural language interface to databases using spatial comprehension. *arXiv preprint arXiv:1908.10917*, 2019.

[Liberty 2005] Jesse Liberty. *Programming C#: Building. NET Applications with C.* " O'Reilly Media, Inc.", 2005.

[Liddy 2001] Elizabeth D Liddy. Natural language processing. 2001.

[Lieberman *et al.* 2006] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. End-user development: An emerging paradigm. In *End user development*, pages 1–8. Springer, 2006.

[Lim *et al.* 2016] Wei Ying Lim, Angela Ong, Lay Lian Soh, and Adam Sufi. Teachers' voices and change: The structure and agency dialectics that shaped teachers' pedagogy toward deep learning. In *Future learning in primary schools*, pages 147–158. Springer, 2016.

[Lin *et al.* 2014] Yuming Lin, Tao Zhu, Hao Wu, Jingwei Zhang, Xiaoling Wang, and Aoying Zhou. Towards online anti-opinion spam: Spotting fake reviews from the review sequence. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 261–264. IEEE, 2014.

[Linz 2011] Peter Linz. *An introduction to formal languages and automata*. Jones & Bartlett Publishers, 2011.

[Lison and Kennington 2016] Pierre Lison and Casey Kennington. Opendial: A toolkit for developing spoken dialogue systems with probabilistic rules. In *Proceedings of ACL-2016 System Demonstrations*, pages 67–72, 2016.

[Littman *et al.* 1986] David C Littman, Jeannine Pinto, Stanley Letovsky, and Elliot Soloway. *Mental models and software maintenance, Papers presented at the first workshop on empirical studies of programmers on Empirical studies of programmers (Norwood, NJ, USA)*, 1986.

[Liu *et al.* 2003] Xia Liu, Lai C Liu, Kai S Koong, and June Lu. An examination of job skills posted on Internet databases: Implications for information systems degree programs. *Journal of Education for Business*, 78(4):191–196, 2003.

[Liu *et al.* 2011] Chen-Chung Liu, Yuan-Bang Cheng, and Chia-Wen Huang. The effect of simulation games on the learning of computational problem solving. *Computers & Education*, 57(3):1907–1918, 2011.

[Liu *et al.* 2017] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[Liu *et al.* 2019] Qian Liu, Bei Chen, Jian-Guang Lou, Ge Jin, and Dongmei Zhang. Fanda: A novel approach to perform follow-up query analysis. *arXiv preprint arXiv:1901.08259*, 2019.

[Lobur *et al.* 2011] Mykhailo Lobur, AnDriy Romanyuk, and Mariana Romanyshyn. Using nltk for educational and scientific purposes. In *2011 11th international conference the experience of designing and application of CAD systems in microelectronics (CADSM)*, pages 426–428. IEEE, 2011.

[Loper and Bird 2002] Edward Loper and Steven Bird. NLTK: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[López *et al.* 2017] Gustavo López, Luis Quesada, and Luis A Guerrero. Alexa vs. Siri vs. Cortana vs. Google assistant: A comparison of speech-based natural user interfaces. In *International Conference on Applied Human Factors and Ergonomics*, pages 241–250. Springer, 2017.

[Lorenz *et al.* 2019] Klara Lorenz, Paul P Freddolino, Adelina Comas-Herrera, Martin Knapp, and Jacqueline Damant. Technology-based tools and services for people with dementia and carers: Mapping technology onto the dementia care pathway. *Dementia*, 18(2):725–741, 2019.

[Lowe *et al.* 2016] Matthew X Lowe, Ryan A Stevenson, Kristin E Wilson, Natasha E Ouslis, Morgan D Barense, Jonathan S Cant, and Susanne Ferber. Sensory processing patterns predict the integration of information held in visual working memory. *Journal of Experimental Psychology: Human Perception and Performance*, 42(2):294, 2016.

[Lu *et al.* 2012] Meiyu Lu, Srinivas Bangalore, Graham Cormode, Marios Hadjieleftheriou, and Divesh Srivastava. A dataset search engine for the research document corpus. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1237–1240. IEEE, 2012.

[Luque *et al.* 2019] Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de las Heras. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231, 2019.

[Luxton-Reilly *et al.* 2018] Andrew Luxton-Reilly, Ibrahim Albluwi, Brett A Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, Claudia Szabo, et al. Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 55–106. ACM, 2018.

[Lye and Koh 2014] Sze Yee Lye and Joyce Hwee Ling Koh. Review on teaching and learning of computational thinking through programming: What is next for k-12? *Computers in Human Behavior*, 41:51–61, 2014.

[Lyons *et al.* 2016] Gabriel Lyons, Vinh Tran, Carsten Binnig, Ugur Cetintemel, and Tim Kraska. Making the case for Query by Voice with Echoquery. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2129–2132. ACM, 2016.

[Madnani 2007] Nitin Madnani. Getting started on natural language processing with Python. *ACM Crossroads*, 13(4):5, 2007.

[Maguluri *et al.* 2019] Lakshmana Phaneendra Maguluri, R Ragupathy, Sita Rama Krishna Buddi, Vamshi Ponugoti, and Tharun Sai Kalimil. Adaptive prediction of spam emails: Using Bayesian inference. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pages 628–632. IEEE, 2019.

[Maier and Größler 2000] Frank H Maier and Andreas Größler. What are we talking about?—a taxonomy of computer simulations to support learning. *System Dynamics Review: The Journal of the System Dynamics Society*, 16(2):135–148, 2000.

[Malan and Leitner 2007] David J Malan and Henry H Leitner. Scratch for budding computer scientists. In *ACM Special Interest Group on Computer Science Education Bulletin*, volume 39, pages 223–227. ACM, 2007.

[Mannila *et al.* 2014] Linda Mannila, Valentina Dagiene, Barbara Demo, Natasa Grgurina, Claudio Mirolo, Lennart Rolandsson, and Amber Settle. Computational thinking in k-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference*, pages 1–29. ACM, 2014.

[Manning *et al.* 1999] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[Manning *et al.* 2010] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[Manning *et al.* 2014] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[Manning 2014] Christopher Manning. Natural language processing. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Citeseer, 2014.

[Manning 2015] Christopher D Manning. Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707, 2015.

[Mannino 2001] Michael Mannino. *Database application development and design*. McGraw-Hill, Inc., 2001.

[Mao and Brown 2007] Ji-Ye Mao and Bradley R Brown. The effectiveness of online task support vs. instructor-led training. *Contemporary Issues in End User Computing*, page 77, 2007.

[Marcus *et al.* 1994] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics, 1994.

[Marcus *et al.* 2019] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Neo: A learned query optimizer. *arXiv preprint arXiv:1904.03711*, 2019.

[Marginean 2017] Anca Marginean. Question answering over biomedical linked data with grammatical framework. *Semantic Web*, 8(4):565–580, 2017.

[Martin 1991] John C Martin. *Introduction to Languages and the Theory of Computation*, volume 4. McGraw-Hill NY, 1991.

[Martin 2003] J.C. Martin. *Introduction to Languages and the Theory of Computation*. McGraw-Hill, New York, 2003.

[Martins *et al.* 2014] Carolina Martins, Tiago Oliveira, and Aleš Popovič. Understanding the Internet banking adoption: A unified theory of acceptance and use of technology and perceived risk application. *International Journal of Information Management*, 34(1):1–13, 2014.

[Mason *et al.* 2016] Raina Mason, Carolyn Seton, and Graham Cooper. Applying cognitive load theory to the redesign of a conventional database systems course. *Computer Science Education*, 26(1):68–87, 2016.

[Maubert and Pinchinat 2013] Bastien Maubert and Sophie Pinchinat. Jumping automata for uniform strategies. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[Mayer and Alexander 2016] Richard E Mayer and Patricia A Alexander. *Handbook of research on learning and instruction*. Taylor & Francis, 2016.

[Mayer 2009] Richard E Mayer. Constructivism as a theory of learning versus constructivism as a prescription for instruction. *Constructivist instruction: Success or failure*, pages 184–200, 2009.

[McCann *et al.* 2016] James McCann, Lea Albaugh, Vidya Narayanan, April Grow, Wojciech Matusik, Jennifer Mankoff, and Jessica Hodgins. A compiler for 3D machine knitting. *ACM Transactions on Graphics*, 35(4):49, 2016.

[McDougle *et al.* 2016] Samuel D McDougle, Richard B Ivry, and Jordan A Taylor. Taking aim at the cognitive side of learning in sensorimotor adaptation tasks. *Trends in cognitive sciences*, 20(7):535–544, 2016.

[McGill 2008] Monica McGill. Critical skills for game developers: an analysis of skills sought by industry. In *Proceedings of the 2008 conference on future play: Research, play, share*, pages 89–96. ACM, 2008.

[McLeod 2003] Gregory McLeod. Learning theory and instructional design. *Learning Matters*, 2(3):35–43, 2003.

[McTear 2002] Michael F McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys (CSUR)*, 34(1):90–169, 2002.

[Mealin and Murphy-Hill 2012] Sean Mealin and Emerson Murphy-Hill. An exploratory study of blind software developers. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 71–74. IEEE, 2012.

[Meduna and Soukup 2017] Alexander Meduna and Ondřej Soukup. *Modern Language Models and Computation: Theory with Applications*. Springer, 2017.

[Meduna and Zemek 2012] Alexander Meduna and Petr Zemek. Jumping finite automata. *International Journal of Foundations of Computer Science*, 23(07):1555–1578, 2012.

[Meduna and Zemek 2014] Alexander Meduna and Petr Zemek. *Regulated Grammars and Automata*. Springer, 2014.

[Meduna 2014] Alexander Meduna. *Formal Languages and Computation: Models and their applications*. Auerbach Publications, 2014.

[Melton 1998] Jim Melton. Database language SQL. In *Handbook on Architectures of Information Systems*, pages 103–128. Springer, 1998.

[Menekse 2019] Muhsin Menekse. The reflection-informed learning and instruction to improve students' academic success in undergraduate classrooms. *The Journal of Experimental Education*, pages 1–17, 2019.

[Merriam *et al.* 2006] Sharan B Merriam, Rosemary S Caffarella, and Lisa M Baumgartner. *Learning in adulthood: A comprehensive guide*. John Wiley & Sons, 2006.

[Metcalfe and Xu 2017] Janet Metcalfe and Judy Xu. Learning from one's own errors and those of others. *Psychonomic Bulletin & Review*, pages 1–7, 2017.

[Metcalfe 2017] Janet Metcalfe. Learning from errors. *Annual review of psychology*, 68:465–489, 2017.

[Meziane *et al.* 2008] Farid Meziane, Nikos Athanasakis, and Sophia Ananiadou. Generating natural language specifications from uml class diagrams. *Requirements Engineering*, 13(1):1–18, 2008.

[Miklody *et al.* 2017] Daniel Miklody, Wendie M Uitterhoeve, Dimitri van Heel, Kerstin Klinkenberg, and Benjamin Blankertz. Maritime cognitive workload assessment. *10.1007/978-3-319-57753-1*, 2017.

[Miller *et al.* 2017] Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017.

[Miller 1995] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[Mira 2008] José Mira Mira. Symbols versus connections: 50 years of artificial intelligence. *Neurocomputing*, 71(4-6):671–680, 2008.

[Mishra and Jain 2016] Amit Mishra and Sanjay Kumar Jain. A survey on question answering systems with classification. *Journal of King Saud University-Computer and Information Sciences*, 28(3):345–361, 2016.

[Mitrovic and Ohlsson 2016] Antonija Mitrovic and Stellan Ohlsson. Implementing CBM: SQL-tutor after fifteen years. *International Journal of Artificial Intelligence in Education*, 26(1):150–159, 2016.

[Mitrovic 1998] Antonija Mitrovic. Learning SQL with a computerized tutor. In *ACM Special Interest Group on Computer Science Education Bulletin*, volume 30, pages 307–311. ACM, 1998.

[Mitrovic 2003] Antonija Mitrovic. An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education*, 13(2-4):173–197, 2003.

[Mitrovic 2012] Antonija Mitrovic. Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User modeling and user-adapted interaction*, 22(1-2):39–72, 2012.

[Mogensen 2009] Torben Ægidius Mogensen. *Basics of Compiler Design*. Torben Ægidius Mogensen, 2009.

[Mohammed *et al.* 2017] Hussein Mohammed, Volker Mäergner, Thomas Konidaris, and H Siegfried Stiehl. Normalised local naïve Bayes nearest-neighbour classifier for offline writer identification. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1013–1018. IEEE, 2017.

[Mohtashami and Scher 2000] Mojgan Mohtashami and Julian M Scher. Application of Bloom's cognitive domain taxonomy to database design. In *Proceedings of ISECON (information systems educators conference)*. Citeseer, 2000.

[Montazemi and Qahri-Saremi 2015] Ali Reza Montazemi and Hamed Qahri-Saremi. Factors affecting adoption of online banking: A meta-analytic structural equation modeling study. *Information & Management*, 52(2):210–226, 2015.

[Mony *et al.* 2014] Manju Mony, Jyothi M Rao, and Manish M Potey. An overview of NLIDB approaches and implementation for airline reservation system. *International Journal of Computer Applications*, 107(5), 2014.

[Moreno-León and Robles 2016] Jesús Moreno-León and Gregorio Robles. Code to learn with scratch? a systematic literature review. In *2016 IEEE Global Engineering Education Conference (EDUCON)*, pages 150–156. IEEE, 2016.

[Morris 1970] Charles W Morris. The pragmatic movement in american philosophy. 1970.

[Mössenböck 2005] Hanspeter Mössenböck. The Compiler Generator Coco/R User Manual. pages 1–33, 2005.

[Mowrer 1960] Orval Mowrer. Learning theory and behavior. 1960.

[MSDN 2017] MSDN. *Regular Expression Language - Quick Reference: Microsoft Developer Network Documentation on the .Net framework, version 4.5, regular expression*. `https://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx`, 2017. Accessed: 2017-05-28.

[Muhammad *et al.* 2016] Aminu Muhammad, Nirmalie Wiratunga, and Robert Lothian. Contextual sentiment analysis for social media genres. *Knowledge-Based Systems*, 108:92–101, 2016.

[Munteanu and Marcu 2005] Dragos Stefan Munteanu and Daniel Marcu. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504, 2005.

[Myalapalli and Shiva 2015] Vamsi Krishna Myalapalli and Muddu Butchi Shiva. An appraisal to optimize SQL queries. In *International Conference on Pervasive Computing)*, pages 1–6. IEEE, 2015.

[Myers 1998] Brad A Myers. A brief history of human computer interaction technology. *interactions*, 5(2):44–54, 1998.

[Nadkarni *et al.* 2011] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, 2011.

[Nagataki *et al.* 2013] Hiroyuki Nagataki, Yoshiaki Nakano, Midori Nobe, Tatsuya Tohyama, and Susumu Kanemune. A visual learning tool for database operation. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, pages 39–40. ACM, 2013.

[Nagy and Tick 2016] Albert Nagy and József Tick. Improving transport management with big data analytics. In *IEEE 14th International Symposium on Intelligent Systems and Informatics*, pages 199–204. IEEE, 2016.

[Najar *et al.* 2014] Amir Shareghi Najar, Antonija Mitrovic, and Bruce M McLaren. Adaptive support versus alternating worked examples and tutored problems: Which leads to better learning? In *International Conference on User Modeling, Adaptation, and Personalization*, pages 171–182. Springer, 2014.

[Najar *et al.* 2016] Amir Shareghi Najar, Antonija Mitrovic, and Bruce M McLaren. Learning with intelligent tutors and worked examples: selecting learning activities adaptively leads to better learning outcomes than a fixed curriculum. *User Modeling and User-Adapted Interaction*, 26(5):459–491, 2016.

[Nakazawa *et al.* 2006] Toshiaki Nakazawa, Kun Yu, Daisuke Kawahara, and Sadao Kurohashi. Example-based machine translation based on deeper NLP. In *International Workshop on Spoken Language Translation (IWSLT) 2006*, 2006.

[Náplava and Straka 2019] Jakub Náplava and Milan Straka. CUNI system for the building educational applications 2019 shared task: Grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 183–190, 2019.

[Narasimhan *et al.* 2016] Vagheesh Narasimhan, Petr Danecek, Aylwyn Scally, Yali Xue, Chris Tyler-Smith, and Richard Durbin. BCFtools/RoH: a hidden Markov model approach for detecting autozygosity from next-generation sequencing data. *Bioinformatics*, 32(11):1749–1751, 2016.

[Nardi *et al.* 2000] Bonnie A. Nardi, Steve Whittaker, and Erin Bradner. Interaction and outeraction: Instant messaging in action. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 79–88, New York, NY, USA, 2000. ACM.

[Nasukawa and Yi 2003] Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77. ACM, 2003.

[Neteler *et al.* 2012] Markus Neteler, M Hamish Bowman, Martin Landa, and Markus Metz. GRASS GIS: A multi-purpose open source GIS. *Environmental Modelling & Software*, 31:124–130, 2012.

[Neumann and Kemper 2015] Thomas Neumann and Alfons Kemper. Unnesting arbitrary queries. *Datenbanksysteme für Business, Technologie und Web (BTW 2015)*, 2015.

[Ng and Low 2004] Hwee Tou Ng and Jin Kiat Low. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 277–284, 2004.

[Ng and Zelle 1997] Hwee Tou Ng and John Zelle. Corpus-based approaches to semantic interpretation in NLP. *AI magazine*, 18(4):45–45, 1997.

[Nguyen *et al.* 2016] Nhu H Nguyen, Zewei Song, Scott T Bates, Sara Branco, Leho Tedersoo, Jon Menke, Jonathan S Schilling, and Peter G Kennedy. Funguild: an open annotation tool for parsing fungal community datasets by ecological guild. *Fungal Ecology*, 20:241–248, 2016.

[Nieuwland *et al.* 2007] Mante S Nieuwland, Marte Otten, and Jos JA Van Berkum. Who are you talking about? tracking discourse-level referential processing with event-related brain potentials. *Journal of Cognitive Neuroscience*, 19(2):228–236, 2007.

[Nihalani *et al.* 2011] Neelu Nihalani, Sanjay Silakari, and Mahesh Motwani. Natural language interface for database: a brief review. *International Journal of Computer Science Issues (IJCSI)*, 8(2):600, 2011.

[Nordlinger and Sadler 2019] Rachel Nordlinger and Louisa Sadler. *Morphology in lexical-functional grammar and head-driven phrase structure grammar*. The Oxford handbook of morphological theory, 2019.

[Norouzifard *et al.* 2008] M Norouzifard, SH Davarpanah, MH Shenassa, et al. Using natural language processing in order to create SQL queries. In *International Conference on Computer and Communication Engineering*, pages 600–604. IEEE, 2008.

[Obaido *et al.* 2018] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli. Generating SQL queries from visual specifications. In *Communications in Computer and Information Science (CCIS)*, pages 315–330, 2018. Springer, Cham, Switzerland. ISBN: 978-3-030-05813-5. URL: https://link.springer.com/chapter/10.1007/978-3-030-05813-5_21 [Switzerland].

[Obaido *et al.* 2019a] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli. Generating narrations of nested SQL queries using context-free grammars. In *the proceedings of IEEE Conference on ICT and Society (ICTAS)*, pages 1–6, 2019. 6th to 9th March, Durban. URL: https://ieeexplore.ieee.org/document/8703620. [South Africa].

[Obaido *et al.* 2019b] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli. Synthesis of SQL queries from narrations. In *the proceedings of the 6th IEEE International Conference on Soft Computing and Machine Intelligence (ISCMI 2019)*, pages 195–201, 2019. ISBN: 978-1-7281-4576-1, November 19th to 20th, 2019 [South Africa].

[Obaido *et al.* 2019c] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli. TalkSQL: A tool for the synthesis of SQL queries from verbal specifications. In *the proceedings of the IEEE International Multidisciplinary Information Technology and Engineering Conference (IMITEC 2019)*, pages 469–478, 2019. November 21st to 22nd, 2019 [South Africa].

[Obare *et al.* 2019] Stephen Obare, Abejide Ade-Ibijola, George Okeyo, and Kennedy Ogada. Jumping finite automata for tweet comprehension. In *the proceedings of the IEEE International Multidisciplinary Information Technology and Engineering Conference (IMITEC 2019),*, pages 361–367, 2019. November 21st to 22nd, 2019 [South Africa].

[Ogden 1986] William C Ogden. Implications of a cognitive model of database query: comparison of a natural language, formal language and direct manipulation interface. *ACM SIGCHI Bulletin*, 18(2):51–54, 1986.

[Ohlsson 2011] Stellan Ohlsson. *Deep learning: How the mind overrides experience*. Cambridge University Press, 2011.

[Okwunma 2018] Nnamdi Ekene Okwunma. *Automatic comprehension of customer queries for feedback generation*, 2018. MSc thesis.

[Omar 2018] Abdul-Majeed Omar. Psychology of language acquisition and EFL teaching methodology: A critical overview. *Gezira Journal of Educational Sciences and Humanities*, 12(2), 2018.

[O'neil 2014] Patrick O'neil. *DATABASE: principles programming performance*. Morgan Kaufmann, 2014.

[Ong *et al.* 2009] Chorng-Shyong Ong, Min-Yuh Day, and Wen-Lian Hsu. The measurement of user satisfaction with question answering systems. *Information & Management*, 46(7):397–403, 2009.

[OpenNLP 2011] Apache OpenNLP. Apache software foundation. *URL http://opennlp.apache.org*, 2011.

[Orikogbo *et al.* 2016] Damilola Orikogbo, Matthias Büchler, and Manuel Egele. Crios: toward large-scale iOS application analysis. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 33–42. ACM, 2016.

[Papangelis *et al.* 2017] Alexandros Papangelis, Panagiotis Papadakos, Margarita Kotti, Yannis Stylianou, Yannis Tzitzikas, and Dimitris Plexousakis. Ld-sds: Towards an expressive spoken dialogue system based on linked-data. *arXiv preprint arXiv:1710.02973*, 2017.

[Papantoniou and Tzitzikas 2019] Katerina Papantoniou and Yannis Tzitzikas. Cs563-qa: A collection for evaluating question answering systems. *arXiv preprint arXiv:1907.01611*, 2019.

[Park *et al.* 2016] Andrew J Park, Brian Beck, Darrick Fletche, Patrick Lam, and Herbert H Tsang. Temporal analysis of radical dark web forum users. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 880–883. IEEE, 2016.

[Pass 2007] Susan Pass. When constructivists Jean Piaget and Lev Vygotsky were pedagogical collaborators: A viewpoint from a study of their communications. *Journal of Constructivist Psychology*, 20(3):277–282, 2007.

[Pazos R *et al.* 2013] Rodolfo A Pazos R, Juan J González B, Marco A Aguirre L, José A Martínez F, and Héctor J Fraire H. Natural language interfaces to databases: an analysis of the state of the art. *Recent Advances on Hybrid Intelligent Systems*, pages 463–480, 2013.

[Peirce 1902] Charles Sanders Peirce. Logic as semiotic: The theory of signs. *Philosophical writings of Peirce*, page 100, 1902.

[Peng *et al.* 2015] Nanyun Peng, Francis Ferraro, Mo Yu, Nicholas Andrews, Jay DeYoung, Max Thomas, Matthew R Gormley, Travis Wolfe, Craig Harman, Benjamin Van Durme, et al. A concrete Chinese NLP pipeline. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 86–90, 2015.

[Pennington 1987] Nancy Pennington. Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive psychology*, 19(3):295–341, 1987.

[Perrin 2003] Dominique Perrin. Automata and formal languages. *Université de Marne-la-vallée*, pages 1–22, 2003.

[Perry 2016] Brain Perry. Advanced guide to the Bloomberg terminal: Bond market functions. *Investopedia Online*, 2016.

[Petre and Blackwell 2007] Marian Petre and Alan F Blackwell. Children as unwitting end-user programmers. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 239–242. IEEE, 2007.

[Pickard and others 2007] Mary J Pickard et al. The new Bloom's taxonomy: An overview for family and consumer sciences. *Journal of Family and Consumer Sciences Education*, 25(1), 2007.

[Pisanski 2014] Katarzyna Pisanski. *Human vocal communication of body size*. PhD thesis, 2014.

[Piwek and Joinson 2016] Lukasz Piwek and Adam Joinson. "what do they snapchat about?" patterns of use in time-limited instant messaging service. *Computers in Human Behavior*, 54:358–367, 2016.

[Politis 2008] Diamanto Politis. The process of entrepreneurial learning: a conceptual framework. In *Entrepreneurial Learning*, pages 66–93. Routledge, 2008.

[Pons *et al.* 2016] Ewoud Pons, Loes MM Braun, MG Myriam Hunink, and Jan A Kors. Natural language processing in radiology: a systematic review. *Radiology*, 279(2):329–343, 2016.

[Ponto 2015] Julie Ponto. Understanding and evaluating survey research. *Journal of the advanced practitioner in oncology*, 6(2):168, 2015.

[Post 1944] Emil L Post. Recursively enumerable sets of positive integers and their decision problems. *bulletin of the American Mathematical Society*, 50(5):284–316, 1944.

[Powell and Wimmer 2015] Loreen M Powell and Hayden Wimmer. Evaluating the effectiveness of self-created student screencasts as a tool to increase student learning outcomes in a hands-on computer programming course. *Information Systems Education Journal*, 13(5):106, 2015.

[Price 2004] Linda Price. Individual differences in learning: Cognitive control, cognitive style, and learning style. *Educational Psychology*, 24(5):681–698, 2004.

[Prior and Lister 2004] Julia Coleman Prior and Raymond Lister. The backwash effect on SQL skills grading. *ACM Special Interest Group on Computer Science Education Bulletin*, 36(3):32–36, 2004.

[Prior 2003] Julia Coleman Prior. Online assessment of SQL query formulation skills. In *Proceedings of the fifth Australasian conference on Computing education-Volume 20*, pages 247–256. Australian Computer Society, Inc., 2003.

[Prior 2014] Julia R Prior. AsseSQL: an online, browser-based SQL skills assessment tool. In *Proceedings of the Innovation & technology in computer science education conference*, pages 327–327. ACM, 2014.

[Pritchard 2017] Alan Pritchard. *Ways of learning: Learning theories for the classroom*. Routledge, 2017.

[Pudaruth *et al.* 2014] Sameerchand Pudaruth, Sandiana Amourdon, and Joey Anseline. Automated generation of song lyrics using CFGs. In *2014 Seventh International Conference on Contemporary Computing (IC3)*, pages 613–616. IEEE, 2014.

[Pyott and Sanders 1991] Sean Pyott and Ian Sanders. Alex: an aid to teaching algorithms. *ACM SIGCSE Bulletin*, 23(3):36–44, 1991.

[Qi and Davison 2009] Xiaoguang Qi and Brian D Davison. Web page classification: Features and algorithms. *ACM computing surveys (CSUR)*, 41(2):12, 2009.

[Qian 2012] Gang Qian. Designing and implementing unsupervised online database labs. *Journal of Computing Sciences in Colleges*, 27(4):30–36, 2012.

[Qin *et al.* 2018] Xuedi Qin, Yuyu Luo, Nan Tang, and Guoliang Li. DeepEye: Visualizing your data by keyword search. In *EDBT*, pages 441–444, 2018.

[Raja 2014] Rao DV Raja. Method and system for automatic switching between chat windows. *Google Patents, US Patent App. 13/738,785*, 2014.

[Rajpurkar *et al.* 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[Ramalingam and Wiedenbeck 1997] Vennila Ramalingam and Susan Wiedenbeck. An empirical study of novice program comprehension in the imperative and object-oriented styles. In *the Seventh Workshop on Empirical studies of programmers*, pages 124–139. ACM, 1997.

[Rangel *et al.* 2014] Rodolfo A Pazos Rangel, Marco A Aguirre, Juan J González, and Juan Martín Carpio. Features and pitfalls that users should seek in natural language interfaces to databases. In *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, pages 617–630. Springer, 2014.

[Rathinam and Sivakumar 2011] Ashok Prasanna Rathinam and Ramani Sivakumar. *System and Method for Managing One or More Databases*, 2011. US Patent App. 12/634,844.

[Rayana and Akoglu 2015] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, pages 985–994. ACM, 2015.

[Reinaldha and Widagdo 2014] Filbert Reinaldha and Tricya E Widagdo. Natural language interfaces to database (NLIDB): Question handling and unit conversion. In *2014 International Conference on Data and Software Engineering (ICODSE)*, pages 1–6. IEEE, 2014.

[Reisner 1977] Phyllis Reisner. Use of psychological experimentation as an aid to development of a query language. *IEEE Transactions on Software Engineering*, (3):218–229, 1977.

[Reiter and Dale 2000] Ehud Reiter and Robert Dale. *Building natural language generation systems*. Cambridge university press, 2000.

[Renaud and H.A.S. 2009] Karen Renaud and H.A.S. Facilitating efficacious transfer of database knowledge and skills. In *Ideas in Teaching, Learning and Assessment of Databases: A Communication of the 7 th International Workshop on Teaching, Learning and Assessment of Databases (TLAD 2009)*, pages 213–224. Springer, 2009.

[Renaud and Van Biljon 2004] Karen Renaud and Judy Van Biljon. Teaching SQL—which pedagogical horse for this course? In *British National Conference on Databases*, pages 244–256. Springer, 2004.

[Renkl 2014] Alexander Renkl. Toward an instructionally oriented theory of example-based learning. *Cognitive science*, 38(1):1–37, 2014.

[Resnick *et al.* 2009] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[Resnick 2013] Mitchel Resnick. Learn to code, code to learn. *EdSurge, May*, 54, 2013.

[Rish and others 2001] Irina Rish et al. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[Ritchie *et al.* 1992] Graeme D Ritchie, Graham J Russell, Alan W Black, and Stephen G Pulman. *Computational morphology: practical mechanisms for the English lexicon*. MIT press, 1992.

[Rizvi *et al.* 2011] Mona Rizvi, Thorna Humphries, Debra Major, Meghan Jones, and Heather Lauzun. A CS0 course using Scratch. *Journal of Computing Sciences in Colleges*, 26(3):19–27, 2011.

[Roark and Johnson 1999] Brian Roark and Mark Johnson. Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 421–428. Association for Computational Linguistics, 1999.

[Roberts *et al.* 2019] Andrew Fiske Roberts, Craig W Stanfill, and Adam Harris Weiss. *Computer screen with visual programming icons*, 2019. US Patent App. 29/582,279.

[Robins *et al.* 2003] Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. *Computer science education*, 13(2):137–172, 2003.

[Roche and Schabes 1997] Emmanuel Roche and Yves Schabes. *Finite-state language processing*. MIT press, 1997.

[Rockoff 2016] Larry Rockoff. *The language of SQL*. Addison-Wesley Professional, 2016.

[Rodrigues *et al.* 2018] Ricardo Rodrigues, Hugo Gonçalo Oliveira, and Paulo Gomes. NLP-port: A pipeline for portuguese NLP (short paper). In *7th Symposium on Languages, Applications and Technologies (SLATE 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[Rodríguez *et al.* 2016] Rosa M Rodríguez, B Bedregal, Humberto Bustince, YC Dong, Bahram Farhadinia, Cengiz Kahraman, L Martínez, Vicenç Torra, YJ Xu, ZS Xu, et al. A position and perspective analysis of hesitant fuzzy sets on information fusion in decision making. towards high quality progress. *Information Fusion*, 29:89–97, 2016.

[Rojit *et al.* 2016] Vidhu Rojit, Sindhu R Pai, Shruthi Kaivalya, and Viraj Kumar. Visual specifications for web-application programming assignments. In *2016 IEEE Eighth International Conference on Technology for Education (T4E)*, pages 50–53. IEEE, 2016.

[Room 2019] Chat Room. Natural language processing. 2019.

[Rosenfeld *et al.* 2017] Ariel Rosenfeld, Abejide Ade-Ibijola, and Sigrid Ewert. Regex parser ii: teaching regular expression fundamentals via educational gaming. In *Annual Conference of the Southern African Computer Lecturers' Association*, pages 99–112. Springer, 2017.

[Roy *et al.* 2019] Pradeep Kumar Roy, Jyoti Prakash Singh, and Snehasish Banerjee. Deep learning to filter SMS spam. *Future Generation Computer Systems*, 2019.

[Ruan *et al.* 2018] Wei Ruan, Naveenkumar Appasani, Katherine Kim, Joseph Vincelli, Hyun Kim, and Won-Sook Lee. Pictorial visualization of emr summary interface and medical information extraction of clinical notes. In *2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pages 1–6. IEEE, 2018.

[Ruohonen 2009] Keijo Ruohonen. Formal languages. *Lecture Notes*, pages 1–94, 2009.

[Rutten *et al.* 2012] Nico Rutten, Wouter R Van Joolingen, and Jan T Van der Veen. The learning effects of computer simulations in science education. *Computers & Education*, 58(1):136–153, 2012.

[Ryan *et al.* 2015] James Owen Ryan, Eric Kaltman, Michael Mateas, and Noah Wardrip-Fruin. What we talk about when we talk about games: Bottom-up game studies using natural language processing. *Proc. FDG*, 2015.

[Saavedra *et al.* 2011] Serguei Saavedra, Kathleen Hagerty, and Brian Uzzi. Synchronicity, instant messaging, and performance among financial traders. *Proceedings of the National Academy of Sciences*, 108(13):5296–5301, 2011.

[Sacerdoti 1977] Earl D Sacerdoti. *Language access to distributed data with error recovery*. Technical report, SRI International Menlo Park CA, Artificial Intelligence Center, 1977.

[Sadiq *et al.* 2004] Shazia Sadiq, Maria Orlowska, Wasim Sadiq, and Joe Lin. SQLator: an online SQL learning workbench. In *ACM Special Interest Group on Computer Science Education Bulletin*, volume 36, pages 223–227. ACM, 2004.

[Sag *et al.* 2002] Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. Multiword expressions: A pain in the neck for NLP. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15. Springer, 2002.

[Sagiroglu and Sinanc 2013] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *International Conference on Collaboration Technologies and Systems*, pages 42–47. IEEE, 2013.

[Saha *et al.* 2016] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Özcan. Athena: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment*, 9(12):1209–1220, 2016.

[Saha *et al.* 2019] Sajal Saha, Mazid Alam, and Smita Dey. A framework for artificially intelligent customized voice response system design using speech synthesis markup. *Intelligent Speech Signal Processing*, page 175, 2019.

[Saini and Rakholia 2016] Jatinderkumar R Saini and Rajnish M Rakholia. On continent and script-wise divisions-based statistical measures for stop-words lists of international languages. *Procedia Computer Science*, 89:313–319, 2016.

[Saini *et al.* 2019] Mayank Saini, Sharad Verma, and Aditi Sharan. Multi-view ensemble learning using rough set based feature ranking for opinion spam detection. In *Advances in Computer Communication and Computational Sciences*, pages 3–12. Springer, 2019.

[Sakaki *et al.* 2011] Takeshi Sakaki, Fujio Toriumi, and Yutaka Matsuo. Tweet trend analysis in an emergency situation. In *Proceedings of the Special Workshop on Internet and Disasters*, page 3. ACM, 2011.

[Sakaki *et al.* 2013] Takeshi Sakaki, Masahide Okazaki, and Yoshikazu Matsuo. Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):919–931, 2013.

[Sander and Wauer 2019] André Sander and Roland Wauer. Integrating terminologies into standard SQL: a new approach for research on routine data. *Journal of biomedical semantics*, 10(1):7, 2019.

[Sandoz *et al.* 2018] Romain Sandoz, Sarah Composto, Sandrine Divorne, Olivier Ertz, and Jens Ingensand. *GeoSQL Journey-A gamified learning experience to introduce (or demystify) geospatial SQL queries*. Technical report, PeerJ Preprints, 2018.

[Sarfraz 2017] Haniya Sarfraz. Strategic leadership development: simplified with Bloom's taxonomy. *Industrial and Commercial Training*, 49(1):40–47, 2017.

[Satyanarayan and Heer 2014] Arvind Satyanarayan and Jeffrey Heer. Lyra: An interactive visualization design environment. In *Computer Graphics Forum*, volume 33, pages 351–360. Wiley Online Library, 2014.

[Savage 2017] Ron Savage. *Database Language SQL (SQL-2003) SQL/Foundation*. https://ronsavage.github.io/SQL/sql-2003-2.bnf.html, 2017. Accessed: 2018-09-30.

[Schlager and Ogden 1986] MS Schlager and William C Ogden. A cognitive model of database querying: a tool for novice instruction. *ACM SIGCHI Bulletin*, 17(4):107–113, 1986.

[Schmitz 2007] Sylvain Schmitz. *Approximating context-free grammars for parsing and verification*. PhD thesis, Université Nice Sophia Antipolis, 2007.

[Schölkopf *et al.* 2002] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[Schulte *et al.* 2010] Carsten Schulte, Tony Clear, Ahmad Taherkhani, Teresa Busjahn, and James H Paterson. An introduction to program comprehension for computer science educators. In *Proceedings of the 2010 ITiCSE working group reports*, pages 65–86. ACM, 2010.

[Schunk 2012] Dale H Schunk. *Learning theories an educational perspective sixth edition*. Pearson, 2012.

[Schweinsberg and Wegner 2017] Kai Schweinsberg and Lutz Wegner. Advantages of complex SQL types in storing XML documents. *Future Generation Computer Systems*, 68:500–507, 2017.

[Scott III 2010] Robert H Scott III. Bloomberg 101. *Journal of Financial Education*, pages 80–88, 2010.

[Sebastiani 2005] Fabrizio Sebastiani. Text categorization. In *Encyclopedia of Database Technologies and Applications*, pages 683–687. IGI Global, 2005.

[Seffah 2015] Ahmed Seffah. *Patterns of HCI design and HCI design of patterns: bridging HCI design and model-driven software engineering*. Springer, 2015.

[Seligman 1970] Martin E Seligman. On the generality of the laws of learning. *Psychological review*, 77(5):406, 1970.

[Selvi *et al.* 2017] S Thamarai Selvi, P Karthikeyan, A Vincent, V Abinaya, G Neeraja, and R Deepika. Text categorization using rocchio algorithm and random forest algorithm. In *2016 Eighth International Conference on Advanced Computing (ICoAC)*, pages 7–12. IEEE, 2017.

[Sengupta and Garg 2019] Subhasree Sengupta and Radhika Garg. Impact of voice-based interaction on learning practices and behavior of children. In *Joint Proceedings of the ACM IUI Workshops, Los Angeles, USA, March 20*, pages 1–3, 2019.

[Serban *et al.* 2016a] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[Serban *et al.* 2016b] Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*, 2016.

[Seuren 2017] Pieter Seuren. *Semantic syntax*. Brill, 2017.

[Seyed-Abbassi 1993] Behrooz Seyed-Abbassi. A SQL project as a learning method in a database course. In *Proceedings of the 1993 conference on Computer personnel research*, pages 291–297. ACM, 1993.

[Seyyedi and Minaei-Bidgoli 2018] Seyyed Hossein Seyyedi and Behrouz Minaei-Bidgoli. Estimator learning automata for feature subset selection in high-dimensional spaces, case study: Email spam detection. *International Journal of Communication Systems*, 31(8):e3541, 2018.

[Shabaz *et al.* 2015] K Shabaz, Jim D O'Shea, Keeley A Crockett, and A Latham. Aneesah: A conversational natural language interface to databases. In *Proceedings of the World Congress on Engineering*, volume 1, 2015.

[Shackelford 1997] Russell L Shackelford. *Introduction to computing and algorithms*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[Shah and Sengupta 2018] Hirav Shah and Amit Sengupta. Designing mobile based computational support for low-literate community health workers. *International Journal of Human-Computer Studies*, 115:1–8, 2018.

[Shah *et al.* 2013] Axita Shah, Jyoti Pareek, Hemal Patel, and Namrata Panchal. NLKBIDB-natural language and keyword based interface to database. In *2013 International conference on advances in computing, communications and informatics (ICACCI)*, pages 1569–1576. IEEE, 2013.

[Shah *et al.* 2017] Philipp Shah, Marc Berges, and Peter Hubwieser. Qualitative content analysis of programming errors. In *Proceedings of the 5th International Conference on Information and Education Technology*, pages 161–166. ACM, 2017.

[Shanmuganathan 2016] Subana Shanmuganathan. Artificial neural network modelling: An introduction. In *Artificial Neural Network Modelling*, pages 1–14. Springer, 2016.

[Sharaff *et al.* 2016] Aakanksha Sharaff, Naresh Kumar Nagwani, and Abhishek Dhadse. Comparative study of classification algorithms for spam email detection. In *Emerging research in computing, information, communication and applications*, pages 237–244. Springer, 2016.

[Shargabi *et al.* 2015] Amal Shargabi, Syed Ahmad Aljunid, Muthukkaruppanan Annamalai, Shuhaida Mohamed Shuhidan, and Abdullah Mohd Zin. Program comprehension levels of abstraction for novices. In *International Conference on Computer, Communications, and Control Technology*, pages 211–215. IEEE, 2015.

[Sharma *et al.* 2019] Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. Natural language understanding with the quora question pairs dataset. *arXiv preprint arXiv:1907.01041*, 2019.

[Sharmila and Sakthi 2018] L Sharmila and U Sakthi. Chronological pattern exploration algorithm for gene expression data clustering and classification. *Wireless Personal Communications*, 102(2):1503–1519, 2018.

[Shawe-Taylor *et al.* 2004] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[Shekarpour *et al.* 2015] Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Sina: Semantic interpretation of user queries for question answering on interlinked data. *Journal of Web Semantics*, 30:39–51, 2015.

[Sheldon 2011] Brian Sheldon. *Cognitive-behavioural therapy: research and practice in health and social care*. Routledge, 2011.

[Sheridan 1959] Peter B Sheridan. The arithmetic translator-compiler of the IBM FORTRAN automatic coding system. *Communications of the ACM*, 2(2):9–21, 1959.

[Shneiderman 1978] Ben Shneiderman. Improving the human factors aspect of database interactions. *ACM Transactions on Database Systems (TODS)*, 3(4):417–439, 1978.

[Shneiderman 2000] Ben Shneiderman. Creating creativity: user interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction*, 7(1):114–138, 2000.

[Shue *et al.* 2009] Craig A Shue, Minaxi Gupta, John J Lubia, Chin Hua Kong, and Asim Yuksel. Spamology: A study of spam origins. In *the 6th Conference on Email and Anti-Spam (CEAS)*, 2009.

[Shum *et al.* 2018] Heung-Yeung Shum, Xiao-dong He, and Di Li. From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26, 2018.

[Siau and Tan 2006] Keng Siau and Xin Tan. Cognitive mapping techniques for user-database interaction. *IEEE transactions on professional communication*, 49(2):96–108, 2006.

[Silva *et al.* 2016] Yasin N Silva, Isadora Almeida, and Michell Queiroz. SQL: From traditional databases to big data. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 413–418. ACM, 2016.

[Singh *et al.* 2013] Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. Automated feedback generation for introductory programming assignments. *ACM Special Interest Group on Programming Languages Notices*, 48(6):15–26, 2013.

[Sinha *et al.* 2015] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246. ACM, 2015.

[Sipser 2006] Michael Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[Socher *et al.* 2012] Richard Socher, Yoshua Bengio, and Christopher D Manning. Deep learning for NLP (without magic). In *Tutorial Abstracts of ACL 2012*, pages 5–5. Association for Computational Linguistics, 2012.

[Soflano *et al.* 2015] Mario Soflano, Thomas M Connolly, and Thomas Hainey. Learning style analysis in adaptive GBL application to teach SQL. *Computers & Education*, 86:105–119, 2015.

[Sokolov *et al.* 2016] Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. Learning structured predictors from bandit feedback for interactive NLP. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1610–1620, 2016.

[Soloway and Ehrlich 1984] Elliot Soloway and Kate Ehrlich. Empirical studies of programming knowledge. *IEEE Transactions on software engineering*, (5):595–609, 1984.

[Sommerville 2015] Ian Sommerville. *Software Engineering*. Addison-Wesley, Boston, MA, USA, 10th edition, 2015.

[Song *et al.* 2015] Dezhao Song, Frank Schilder, Charese Smiley, Chris Brew, Tom Zielund, Hiroko Bretz, Robert Martin, Chris Dale, John Duprey, Tim Miller, et al. TR Discover: A natural language interface for querying and analyzing interlinked datasets. In *International Semantic Web Conference*, pages 21–37. Springer, 2015.

[Sonnhammer *et al.* 1998] Erik LL Sonnhammer, Gunnar Von Heijne, Anders Krogh, et al. A hidden Markov model for predicting transmembrane helices in protein sequences. In *Ismb*, volume 6, pages 175–182, 1998.

[Sowell *et al.* 2009] Benjamin Sowell, Alan Demers, Johannes Gehrke, Nitin Gupta, Haoyuan Li, and Walker White. From declarative languages to declarative processing in computer games. *arXiv preprint arXiv:0909.1770*, 2009.

[Soylu *et al.* 2016] Ahmet Soylu, Martin Giese, Ernesto Jimenez-Ruiz, Guillermo Vega-Gorgojo, and Ian Horrocks. Experiencing optiqueVQS: a multi-paradigm and ontology-based visual query system for end users. *Universal Access in the Information Society*, 15(1):129–152, 2016.

[Soylu *et al.* 2017] Ahmet Soylu, Martin Giese, Ernesto Jimenez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, and Ian Horrocks. Ontology-based end-user visual query formulation: Why, what, who, how, and which? *Universal Access in the Information Society*, 16(2):435–467, 2017.

[Srinivasa-Desikan 2018] Bhargav Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[Srinivasan and Brown 2002] Savitha Srinivasan and Eric Brown. Is speech recognition becoming mainstream? *Computer*, (4):38–41, 2002.

[Starr *et al.* 2008] Christopher W Starr, Bill Manaris, and RoxAnn H Stalvey. Bloom's taxonomy revisited: specifying assessable learning objectives in computer science. In *ACM SIGCSE Bulletin*, volume 40, pages 261–265. ACM, 2008.

[Staykova 2014] Kamenka Staykova. Natural language generation and semantic technologies. *Cybernetics and Information Technologies*, 14(2):3–23, 2014.

[Stein *et al.* 2019] Roger Alan Stein, Patricia A Jaques, and João Francisco Valiati. An analysis of hierarchical text classification using word embeddings. *Information Sciences*, 471:216–232, 2019.

[Storey 2005] M-A Storey. Theories, methods and tools in program comprehension: Past, present and future. In *13th International Workshop on Program Comprehension (IWPC'05)*, pages 181–191. IEEE, 2005.

[Storey 2006] Margaret-Anne Storey. Theories, tools and research methods in program comprehension: past, present and future. *Software Quality Journal*, 14(3):187–208, 2006.

[Straka and Straková 2017] Milan Straka and Jana Straková. Tokenizing, PoS tagging, lemmatizing and parsing UD 2.0 with UDpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, 2017.

[Sturmfels *et al.* 2018] Pascal Sturmfels, Saige Rutherford, Mike Angstadt, Mark Peterson, Chandra Sripada, and Jenna Wiens. A domain guided CNN architecture for predicting age from structural brain images. *arXiv preprint arXiv:1808.04362*, 2018.

[Sujatha *et al.* 2012] B Sujatha, Dr S Viswanadha Raju, and Humera Shaziya. A survey of natural language interface to database management system. *International Journal of Science and Advance Technology*, 2(6):56–61, 2012.

[Sun *et al.* 2017a] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[Sun *et al.* 2017b] Shiliang Sun, Chen Luo, and Junyu Chen. A review of natural language processing techniques for opinion mining systems. *Information fusion*, 36:10–25, 2017.

[Suranauwarat 2017] Sukanya Suranauwarat. An approach to solving technical difficulties facing non-CS students in a database class. *International Journal of Modern Education and Computer Science*, 9(2):14, 2017.

[Sweller 1988] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.

[Sweller 2006] John Sweller. The worked example effect and human cognition. *Learning and instruction*, 16(2):165–169, 2006.

[Sweta and Lal 2016] Soni Sweta and Kanhaiya Lal. Learner model for automatic detection of learning style using FCM in adaptive e-learning system. *IOSR Journal of Engineering*, 18(2):18–24, 2016.

[Taipalus *et al.* 2018] Toni Taipalus, Mikko Siponen, and Tero Vartiainen. Errors and complications in SQL query formulation. *ACM Transactions on Computing Education (TOCE)*, 18(3):15, 2018.

[Taleb 2012] Nassim Nicholas Taleb. *Antifragile: Things That Gain from Disorder (Incerto Book 3)*. Random House, New York, NY, USA, 2012.

[Tang *et al.* 2016] Bo Tang, Haibo He, Paul M Baggenstoss, and Steven Kay. A Bayesian classification approach using class-specific features for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1602–1606, 2016.

[Tang 2013] Yichuan Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013.

[Thomas 2014] Jenny A Thomas. *Meaning in interaction: An introduction to pragmatics*. Routledge, 2014.

[Thompson *et al.* 2008] Errol Thompson, Andrew Luxton-Reilly, Jacqueline L Whalley, Minjie Hu, and Phil Robbins. Bloom's taxonomy for CS assessment. In *Proceedings of the tenth conference on Australasian computing education-Volume 78*, pages 155–161. Australian Computer Society, Inc., 2008.

[Thurley and Dennick 2008] P Thurley and R Dennick. Problem-based learning and radiology. *Clinical radiology*, 63(6):623–628, 2008.

[Tidwell 1999] Jenifer Tidwell. *Common ground: A pattern language for human-computer interface design*, 1999.

[Tidwell 2002] Jenifer Tidwell. UI patterns and techniques. *Zdroj: www. mit. edu/˜ jtidwell*, 2002.

[Tidwell 2010] Jenifer Tidwell. *Designing interfaces: Patterns for effective interaction design*. " O'Reilly Media, Inc.", 2010.

[Tittizer and Ramamoorthy 2006] Abigail Tittizer and Venkatesan Ramamoorthy. *Systems and methods for managing data associated with computer code*, 2006. US Patent App. 11/379,390.

[Tobin 2012] Kenneth G Tobin. Constructivism as a referent for teaching and learning. In *The practice of constructivism in science education*, pages 19–38. Routledge, 2012.

[Tobon-Mejia *et al.* 2011] Diego A Tobon-Mejia, Kamal Medjaher, Noureddine Zerhouni, and Gérard Tripot. Hidden Markov models for failure diagnostic and prognostic. In *2011 Prognostics and System Health Managment Confernece*, pages 1–8. IEEE, 2011.

[Torabi *et al.* 2015] Zahra S Torabi, Mohammad H Nadimi-Shahraki, and Akbar Nabiollahi. Efficient support vector machines for spam detection: a survey. *International Journal of Computer Science and Information Security*, 13(1):11, 2015.

[Trask 2004] Robert Lawrence Trask. *A dictionary of phonetics and phonology*. Routledge, 2004.

[Tratz *et al.* 2007] Stephen Tratz, Antonio Sanfilippo, Michelle Gregory, Alan Chappell, Christian Posse, and Paul Whitney. Pnnl: A supervised maximum entropy approach to word sense disambiguation. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 264–267. Association for Computational Linguistics, 2007.

[Truong 2016] Huong May Truong. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities. *Computers in Human Behavior*, 55:1185–1193, 2016.

[Tsarfaty *et al.* 2010] Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. Statistical parsing of morphologically rich languages (SPMRL): what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12. Association for Computational Linguistics, 2010.

[Turchin *et al.* 2006] Alexander Turchin, Nikheel S Kolatkar, Richard W Grant, Eric C Makhni, Merri L Pendergrass, and Jonathan S Einbinder. Using regular expressions to abstract blood pressure and treatment intensification information from the text of physician notes. *Journal of the American Medical Informatics Association*, 13(6):691–695, 2006.

[Twain 2013] Mark Twain. Morphology: the words of language. 2013.

[Ullman and Aho 1977] Jeffrey D Ullman and Alfred V Aho. Principles of compiler design. *Reading: Addison Wesley*, 1977.

[Utama *et al.* 2017] Prasetya Utama, Nathaniel Weir, Carsten Binnig, and Ugur Çetintemel. Voice-based data exploration: Chatting with your database. In *Proceedings of the 2017 workshop on Search-Oriented Conversational AI*, pages 1–6, 2017.

[Utama *et al.* 2018] Prasetya Utama, Nathaniel Weir, Fuat Basik, Carsten Binnig, Ugur Cetintemel, Benjamin Hättasch, Amir Ilkhechi, Shekar Ramaswamy, and Arif Usta. An end-to-end neural natural language interface for databases. *arXiv preprint arXiv:1804.00401*, 2018.

[Van den Brand *et al.* 2005] Mark Van den Brand, P-E Moreau, and Jurgen Vinju. Generator of efficient strongly typed abstract syntax trees in Java. *IEE Proceedings-Software*, 152(2):70–78, 2005.

[Van Gog *et al.* 2011] Tamara Van Gog, Liesbeth Kester, and Fred Paas. Effects of worked examples, example-problem, and problem-example pairs on novices' learning. *Contemporary Educational Psychology*, 36(3):212–218, 2011.

[Van Kasteren *et al.* 2010] TLM Van Kasteren, Gwenn Englebienne, and Ben JA Kröse. Activity recognition using semi-Markov models on real world smart home datasets. *Journal of ambient intelligence and smart environments*, 2(3):311–325, 2010.

[Van Wanrooij and Pras 2010] Ward Van Wanrooij and Aiko Pras. Filtering spam from bad neighborhoods. *International Journal of Network Management*, 20(6):433–444, 2010.

[Van Welie and Van der Veer 2003] Martijn Van Welie and Gerrit C Van der Veer. Pattern languages in interaction design: Structure and organization. In *Proceedings of interact*, volume 3, pages 1–5, 2003.

[Verenna *et al.* 2018] Anne-Marie A Verenna, Kim A Noble, Helen E Pearson, and Susan M Miller. Role of comprehension on performance at higher levels of Bloom's taxonomy: Findings from assessments of healthcare professional students. *Anatomical sciences education*, 11(5):433–444, 2018.

[Verma *et al.* 2019] Amit Verma, Kirill M Yurov, Peggy L Lane, and Yuliya V Yurova. An investigation of skill requirements for business and data analytics positions: A content analysis of job advertisements. *Journal of Education for Business*, 94(4):243–250, 2019.

[Vermunt and Donche 2017] Jan D Vermunt and Vincent Donche. A learning patterns perspective on student learning in higher education: state of the art and moving forward. *Educational Psychology Review*, 29(2):269–299, 2017.

[Von Mayrhauser and Vans 1993] Anneliese Von Mayrhauser and A Marie Vans. From program comprehension to tool requirements for an industrial environment. In *[1993] IEEE Second Workshop on Program Comprehension*, pages 78–86. IEEE, 1993.

[Von Mayrhauser and Vans 1995] Anneliese Von Mayrhauser and A Marie Vans. Program comprehension during software maintenance and evolution. *Computer*, 28(8):44–55, 1995.

[Voorhees and Harman 1999] Ellen M Voorhees and Donna Harman. The Text REtrieval Conference (trec): History and plans for TREC-9. In *ACM SIGIR Forum*, volume 33, pages 12–15. ACM, 1999.

[Voulodimos *et al.* 2018] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[Wallace 2009] Richard S Wallace. The anatomy of ALICE. In *Parsing the Turing Test*, pages 181–210. Springer, 2009.

[Walter *et al.* 2012] Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. Evaluation of a layered approach to question answering over linked data. In *International Semantic Web Conference*, pages 362–374. Springer, 2012.

[Wang *et al.* 2006] Kevin Wang, Corey McCaffrey, Daniel Wendel, and Eric Klopfer. 3D game design with programming blocks in StarLogo TNG. In *Proceedings of the 7th international conference on Learning sciences*, pages 1008–1009. International Society of the Learning Sciences, 2006.

[Wang *et al.* 2016] Xinggang Wang, Buyun Sheng, Liang Xue, and Zheng Xiao. Classification of customer requirements on map reduce-based naive Bayes. In *2016 IEEE International Conference on Big Data Analysis (ICBDA)*, pages 1–4. IEEE, 2016.

[Wang *et al.* 2017a] Chenglong Wang, Alvin Cheung, and Rastislav Bodik. Interactive query synthesis from input-output examples. In *Proceedings of the ACM International Conference on Management of Data*, pages 1631–1634. ACM, 2017.

[Wang *et al.* 2017b] Chenglong Wang, Alvin Cheung, and Rastislav Bodik. Synthesizing highly expressive SQL queries from input-output examples. In *ACM Special Interest Group on Programming Languages Notices*, volume 52, pages 452–466. ACM, 2017.

[Ward 2015] Patrick T Ward. *Teaching of SQL Through a Game*. PhD thesis, 2015.

[Ward 2016] Gregory L Ward. *The semantics and pragmatics of preposing*. Routledge, 2016.

[Warner and Hirschberg 2012] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media*, pages 19–26. Association for Computational Linguistics, 2012.

[Warnke 2009] Elizabeth Warnke. Technical writing for software documentation writers: A textbook on process and product. 2009.

[Warren and Pereira 1982] D Warren and F Pereira. An efficient for interpreting easily adaptable system natural language queries. *American Journal*, 8(3-4):110–122, 1982.

[Warth *et al.* 2008] Alessandro Warth, Takashi Yamamiya, Yoshiki Ohshima, and Scott Wallace. Toward a more scalable end-user scripting language. In *Sixth International Conference on Creating, Connecting and Collaborating through Computing*, pages 172–178. IEEE, 2008.

[Watson *et al.* 2011] Christopher Watson, Frederick WB Li, and Rynson WH Lau. Learning programming languages through corrective feedback and concept visualisation. In *International Conference on Web-Based Learning*, pages 11–20. Springer, 2011.

[Weizenbaum and others 1966] Joseph Weizenbaum et al. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[Weston *et al.* 2012] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

[Wiedenbeck 1986] Susan Wiedenbeck. Beacons in computer program comprehension. *International Journal of Man-Machine Studies*, 25(6):697–709, 1986.

[Wilcock 2009] Graham Wilcock. Text annotation with openNLP and Uima. 2009.

[Wilcocks and Sanders 1994] Derek Wilcocks and Ian Sanders. Animating recursion as an aid to instruction. *Computers & Education*, 23(3):221–226, 1994.

[Wilhelm *et al.* 2013] Reinhard Wilhelm, Helmut Seidl, and Sebastian Hack. *Compiler design: Syntactic and semantic analysis*. Springer Science & Business Media, 2013.

[Wilson *et al.* 2010] Dale-Marie Wilson, Aqueasha M Martin, and Juan E Gilbert. How may i help you'- spoken queries for technical assistance. In *Proceedings of the 48th Annual Southeast Regional Conference*, page 43. ACM, 2010.

[Wilson 2018] Leslie Owen Wilson. Anderson and krathwohl–Bloom's taxonomy revised. *Accessed online: https://thesecondprinciple. com/teaching-essentials/beyond-bloom-cognitive-taxonomy-revised*, 2018.

[Winograd 1973] Terry Winograd. A procedural model of language understanding. 1973.

[Winthrop and McGivney 2015] Rebecca Winthrop and Eileen McGivney. Why wait 100 years? bridging the gap in global education. *The Brookings Institution*, 2015.

[Wolfe and Kolb 1984] Donald M Wolfe and David A Kolb. Career development, personal growth and experiential learning. *D. Kolb, IM Rubin and JM McIntyre op. cit*, 1984.

[Wood *et al.* 2019] Alexander Wood, Vladimir Shpilrain, Kayvan Najarian, and Delaram Kahrobaei. Private naive Bayes classification of personal biomedical data: Application in cancer data analysis. *Computers in biology and medicine*, 105:144–150, 2019.

[Woods 1972] William Woods. The lunar sciences natural language information system. *BBN report*, 1972.

[Woods 1973] William A Woods. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450. ACM, 1973.

[Woods 1978] William A Woods. Semantics and quantification in natural language question answering. In *Advances in computers*, volume 17, pages 1–87. Elsevier, 1978.

[Woolf 2010] Beverly Park Woolf. *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, 2010.

[Wudaru *et al.* 2019] Vishal Wudaru, Nikhil Koditala, Aruneswara Reddy, and Radhika Mamidi. Question answering on structured data using NLIDB approach. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 1–4. IEEE, 2019.

[Xie *et al.* 2008] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: signatures and characteristics. *ACM SIGCOMM Computer Communication Review*, 38(4):171–182, 2008.

[Xu *et al.* 2017] Xiaojun Xu, Chang Liu, and Dawn Song. SQLnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*, 2017.

[Xu *et al.* 2018] Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. SQL-to-text generation with graph-to-sequence model. *arXiv preprint arXiv:1809.05255*, 2018.

[Xu 2018] Shuo Xu. Bayesian naïve Bayes classifiers to text classification. *Journal of Information Science*, 44(1):48–59, 2018.

[Yaghmazadeh *et al.* 2017a] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. SQLizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):63, 2017.

[Yaghmazadeh *et al.* 2017b] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. Type-and content-driven synthesis of SQL queries from natural language. *arXiv preprint arXiv:1702.01168*, 2017.

[Yang *et al.* 2018] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[Yassine *et al.* 2017] Alaeeddine Yassine, Driss Chenouni, Mohammed Berrada, and Ahmed Tahiri. A serious game for learning C programming language concepts using Solo taxonomy. *International Journal of Emerging Technologies in Learning (iJET)*, 12(03):110–127, 2017.

[Yathongchai *et al.* 2017] Wilairat Yathongchai, Jitimon Angskun, and Chun Che FUNG. An ontology model for developing a SQL personalized intelligent tutoring system. *Naresuan University Journal: Science and Technology*, 25(4):88–96, 2017.

[Yu 2012] Sheng Yu. Regular languages. *Handbook of Formal Languages: Volume 1 Word, Language, Grammar*, page 41, 2012.

[Yuan *et al.* 2019] Chi Yuan, Patrick B Ryan, Casey Ta, Yixuan Guo, Ziran Li, Jill Hardin, Rupa Makadia, Peng Jin, Ning Shang, Tian Kang, et al. Criteria2query: a natural language interface to clinical databases for cohort definition. *Journal of the American Medical Informatics Association*, 26(4):294–305, 2019.

[Zamfir *et al.* 2019] Vlad-Andrei Zamfir, Mihai Carabas, Costin Carabas, and Nicolae Tapus. Systems monitoring and big data analysis using the elasticsearch system. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pages 188–193. IEEE, 2019.

[Zhang and Sun 2013] Sai Zhang and Yuyin Sun. Automatically synthesizing SQL queries from input-output examples. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 224–234. IEEE, 2013.

[Zhang *et al.* 2018] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. Understanding and mitigating the security risks of voice-controlled third-party skills on Amazon Alexa and Google Home. *arXiv preprint arXiv:1805.01525*, pages 1–16, 2018.

[Zhang *et al.* 2019a] Hong Zhang, Hanshuo Wei, Yeye Tang, and Qiumei Pu. Research on classification of scientific and technological documents based on naive Bayes. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, pages 327–331. ACM, 2019.

[Zhang *et al.* 2019b] Yang Zhang, Tianyuan Liu, Liqun Chen, Jinxurong Yang, Jiayi Yin, Yuncong Zhang, Zhixi Yun, Hao Xu, Lin Ning, Fengbiao Guo, et al. Riscoper: a tool for RNA–RNA interaction extraction from the literature. *Bioinformatics*, 2019.

[Zhao *et al.* 2009] Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum. BotGraph: Large scale spamming botnet detection. In *NSDI*, volume 9, pages 321–334, 2009.

[Zheng *et al.* 2013] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. Deep learning for Chinese word segmentation and PoS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, 2013.

[Zheng *et al.* 2017] Weiguo Zheng, Hong Cheng, Lei Zou, Jeffrey Xu Yu, and Kangfei Zhao. Natural language question/answering: Let users talk with the knowledge graph. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 217–226. ACM, 2017.

[Zoph *et al.* 2016] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*, 2016.

COLOPHON

This work is based on a research supported by the Council for Scientific and Industrial Research of South Africa (CSIR). Any opinion, findings and conclusions or recommendations expressed in this material are those of the author and therefore, the CSIR does not accept liability in regard thereto.

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyXat: https://bitbucket.org/amiede/classicthesis/. The author of this thesis has made many modifications to the `classicthesis` template to arrive at this final version

---

SQL Comprehension and Synthesis

George Rabeshi Obaido | ***2020***
rabeshi.george@gmail.com