NOVA IMS Information Management School

MDSAA

Master's Degree Program in

Data Science and Advanced Analytics

A Neural Rule-Based Model for Database Exploration

The combination of rule-based methods with pre-trained embedding models for text to SQL task.

Yuriy Perezhohin

Thesis

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

A NEURAL RULE-BASED MODEL FOR DATABASE EXPLORATION

by

Yuriy Perezhohin

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

Supervisor: Mauro Castelli

Co-Supervisor*:* Fernando Augusto Junqueira Peres

July 2023

Declaration of Integrity

I declare that I have carried out this academic work with integrity. I confirm that I have not resorted to plagiarism or any other form of misuse of information or falsification of results during the process of preparing this work. I also declare that I am aware of the Rules of Conduct and the Code of Honor of the NOVA Information Management School.

Yuriy Perezhohin

Lisbon, 10 of July of 2023

Acknowledgements

First of all, I would like to say Thank You to both my supervisors Mauro Castelli and Fernando Peres, for their important support and numerous opportunities provided. Without them, I would not have learned so many things in this difficult field and grown as a person. Thank you, you are the best!

I would like to thank my parents for the important moral and physical support that they provided, it was very important for me as a son and I wouldn't be where I am now without them. Thank you I made it!

I would like to express my gratitude to the "alface fresca e fofa" Sofia for the important advice and encouragement that she provided during my journey. Additionally would like to thank her husband João for the opportunity to stay in his house and for the warm accommodation.

To my loved one Susana, I am incredibly grateful for having you by my side. Thank you for all the unwavering support, for the late nights and long days which you dedicated to me. I want you to know that without you the "proposed algorithm" would not be made. I hope that we grow old and rich as we are planning and that everybody knows who is the PhD couple.

A big Thank You to the "Deus do Trovao", Nuno for every piece of advice, support, encouragement and more important for believing in me until the very end. I hope our long-lasting friendship continues to grow steadily and prepare yourself for the future because it is very bright.

I would like to thank Augusto a friend, colleague and business partner, for all the patience, encouragement and support you provided. Continue to contaminate me with positivism and remain the same person you are. "You rock"

Finally, I want to express my gratitude to every professor at NOVA University that I have crossed during my journey. I want you to know that all of you contributed to this work with the provided knowledge and experience.

Abstract

The problem of accessing Relational Databases in plain language is a difficult task that has been explored vastly in the last decades. Many approaches emerged, from machine learning methods to deep learning ones. However, all of them present several gaps, such as a lack of accuracy, poor domain generalisability, and high computational training costs. This study investigates the possibility of creating an alternative approach free of the training phase. Consequently, combining several linguistic rules with pre-trained embedding models, a hand-written vocabulary, and a set of structuring rules to replicate correct queries from text questions. This approach is tested on the pre-selected databases with different domains as also on the entire benchmark. The performance is accessed by two different metrics, the execution accuracy and the set of clauses that are correctly predicted. Besides these metrics, it is also compared the computational time needed to run the whole experiment given distinct embedding models and the similarity metrics. The experiment identified the most effective embedding model for the proposed algorithm and demonstrated sustainable accuracy within different databases. This study showcases an alternative approach to this challenging task and establishes a broad future work to follow.

Keywords

Natural Language Processing, text-to-SQL, Computational Linguistics, Rule-Based, Sentence Embeddings

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

ALBERT A Little BERT

BERT Bidirectional Encoder Representations from Transformers

BOW Bag-of-Words

CBOW Common Bag of Words

CRUD Create Read Update Delete

csv comma-separated values

GNN Graph Neural Network

IR Information Retrieval

json JavaScript Object Notation

LSTM Long Short Term Memory

NLG Natural Language Generation

NLI Natural Language Interfaces

NLIDB Natural Language Interface for Data Bases

NLP Natural Language Processing

NLU Natural Language Understanding

NN Neural Networks

OOV Out-of-Vocabulary

PLMs Pre-Trained Language Models

POS Part-of-Speech

RDBMS Relational Database Management Systems

RoBERTa Robustly Optimized BERT Pretraining Approach

seq2seq Sequence to Sequence

seq2set Sequence to Set

SQL Structured Query Language

TF-IDF Term Frequency-Inverse Document Frequency

Chapter 1

Introduction

Natural Language Processing (NLP) field has been vastly explored since the end of World War II. This field focuses on the interaction between computers and humans, enabling the understanding and generation of human language. One of its sub-fields is Natural Language Interfaces (NLI), which works with two groups of data, unstructured and structured (Majhadi and Machkour, 2021). NLI for structured data started drawing increased attention since the 1990s when Relational Database Management Systems (RDBMS) surged and began to be used worldwide by many business companies. To extract the information from these databases, end users like Consultants or Business Analysts needed to be familiarised with Structured Query Language (SQL) to access data (e.g. Select, Filter, Having Clauses). Therefore, users without this knowledge could not extract valuable information from databases. This requirement difficulted access to information, especially for non-technical experts. Recently, especially in the last two decades, the Natural Language Interface for Data Bases (NLIDB) became a hot research topic, and many text-to-SQL models were produced using different approaches to satisfy the need to retrieve the necessary information from the databases. With the fast-paced rhythm of the currently evolving Artificial Intelligence (AI) technology, the NLIDB are expanding to several domains to satisfy business needs. However, despite the progress in this area, several opportunities and necessities exist to improve NLIDB to meet users' expectations and tackle the difficulties with language context and understanding.

A competent NLIDB system for an inexperienced user, especially without SQL knowledge, must be capable of interacting with data and dealing with several challenges as answering questions with minimal information. This means that the currently developed NLIDB poses several problems:

- **Natural Language Ambiguity** - The most significant problem until now is the unexplicit requests, which can vary from user to user. The capacity to process and understand can significantly decrease if the whole context is not presented in the user's question (Abbas et al., 2022). Current NLIDB systems are trained on benchmarks that do not tackle a natural language's ambiguity, which means the models mainly understand clear and specific questions.

- **Request Incompleteness** - One major problem interacting with users is the incompletion of their questions. A question should have complete and clear information. Otherwise, the system would struggle to understand the intent and produce non-working queries or even wrong answers.

- **Cross Domain Adoptability** - The majority of NLIDB systems, which were developed in the real-world scenario, are constrained to specific use cases to yield higher results (e.g. LUNAR Woods (1973)). Even with the development of new and innovative deep-learning techniques and the availability of cross-domain datasets to train the models, these systems are still very shallow to be used in the industrial scope. Consequently, the database domain of the industrial field mainly differs from the academic ones on which the models are trained, and the quantity of training examples is insufficient. Additionally, resulting in poor generalisability of the current text-to-SQL models (Qin et al., 2022).

- **Accessibility** - At the moment foremost, text-to-SQL models are trained via deep learning techniques, which makes them inaccessible and not interpretable. The model consisted of many layers of Neural Networks (NN) that do not give access to the user to change, if necessary, any components acting as a "*black-box*". Changing anything or fine-tuning the model for a specific case would require collecting high-quality training data and waiting for the fine-tuning process.

Given the abovementioned issues, creating a capable NLIDB where a business user could navigate their databases through natural language is problematic, leaving a big gap for research and innovation.

1.1 Objectives

The main goal of this Master Thesis is to investigate alternatives to create a different approach for a text-to-SQL task free of training. Our guiding hypothesis is that combining rule-based techniques with handwritten vocabulary and pre-trained models fine-tuned for semantic search can correctly translate a question into a SQL code. The core idea is to create a zero-shot NLIDB capable of generalising to new domains without any prior training based only on linguistic and handwritten rules. As of today, the only zero-shot models capable of tackling the difficult task of text-to-SQL and which are present on the academic benchmarks are made by technological companies. These models, such as *GPT 3* (Liu et al., 2023), use a prompting technique to pass all the database information and the question to translate the text into SQL code. However, these models take a huge amount of time and resources to train. Given that and the challenges identified by Quamar et al. (2022), Majhadi and Machkour (2021) and Abbas et al. (2022), this work focuses on answering the following research questions:

- **RQ1** - Would a combination of several linguistic rules together with pre-trained models for semantic search correctly identify which column and table is necessary to select from a question within a database?

- **RQ2** - Is it possible to correctly structure SQL queries based only on rule-based techniques in the context of free training?

The main contribution of this work to the academic field is that it proposes a novel approach which is more resource-efficient and domain flexible. This approach has significant potential for improvement and advancement through the line of research chosen

in this work. Togelius and Yannakakis (2023) have identified that the current pace of AI advances made by the private sector, which invests enormous amounts of resources, is constraining academic researchers from competing on the global scale. Furthermore, the absence of extensive training requirements opens up opportunities for researchers to explore innovative ideas and techniques, which could lead to potential breakthroughs in solving the text-to-SQL problem.

1.2 Document Structure

The introductory chapter of this dissertation provides context for the problem being addressed and presents the main objective and contributions of the research. The subsequent chapters are structured as follows:

- **Chapter 2** defines and explains all vital concepts for understanding the problem and the proposed solution.
- **Chapter 3** presents the majority of work done in NLIDB and also overviews the word embeddings.
- **Chapter 4** defines a methodology used to create and test the NLIDB in this work.
- **Chapter 5** presents the text-to-SQL algorithm for the databases and the earlier version for one table only.
- **Chapter 6** summarises the results of the proposed algorithm.
- **Chapter 7** discusses the conclusions and limitations of this work and presents the future work

Chapter 2

Background

2.1 Natural Language Processing

In recent years NLP has earned large research public as it is a discipline within the Artificial Intelligence field. This area of research performs various human-like language processing activities through extensive computational techniques such as transforming text into structured data, using one or more levels of linguistic analysis on natural utterances (Liddy, 2001). NLP consists of two main subfields: Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLU involves transcribing any given expression into a machine understandable action by extracting the metadata from it. This extraction happens by analysing seven interdependent levels used to bring out the meaning of the natural language. In Table 2.1, all the levels are depicted in a more detailed way (Feldman, 1999; Liddy, 1998). As the name suggests, NLG is the process of generating written or spoken text, and its application follows by determining what should be communicated, deciding how the answer set should be structured and generating a grammatically correct expression (Reiter and Dale, 1997). The aforementioned fields are, in some tasks, intrinsically related as there is a need to understand input text, convert it into structured data and generate meaningful sentences.

Table 2.1: Seven levels of linguistic structure

Level	Function
Phonetic	Distinguishing words by sound
Morphological	Analysing structure of words as roots, suffixes and prefixes
Lexical	Analysing part of speech and lexical meaning of the word
Syntactic	Analysing grammar and structure of sentences
Semantic	Analysing meaning of words and sentences
Discourse	Analysing the structure of the sentence to the general context
Pragmatic	Analysing the information that is known from general context

2.1.1 Natural Language Understanding

The goals of NLP and NLU are very similar, both seeking to make sense of unstructured data. The former process tries to process and facilitate communication between computers

and humans, while the second makes sense of it. Many tasks such as Machine Translation, Automated Reasoning or Question Answering belong to the NLU field. Multiple alternative interpretations could mislead those tasks as ambiguity is inherent in human language. Although every interdependent level could generate ambiguity, these are the three most common:

- **Lexical Ambiguity** - different meanings of the same word (e.g., *fan* could represent a tool or a supporter).
- **Syntactic Ambiguity** - one sentence can be interpreted in several ways because of its syntactic structure. For example, the word *watch* can be interpreted as a *Noun* or as *Verb* (MacDonald et al., 1994).
- **Semantic Ambiguity** - words can refer to more than one concept. The example *Look at the dog with one eye* can mean the dog has one eye or the action of looking with one eye closed(Rodd et al., 2002).

2.1.2 Natural Language Generation

Creating natural language utterances on purpose is closely related to understanding them, where both generation and understanding field shares a theoretical background to accomplish their complex tasks (Reiter and Dale, 1997). However, there is a certain difference in the tasks done by NLG as it enables the writing capabilities of a machine, thus allowing the generation of automatic reports, chat-bots, creative writing and computational humour. These auto-generating systems are possible by giving the right data for ingestion, so the machine can understand what to generate. Therefore one big question is what data is needed to compose an utterance (McDonald, 1993). With that question, some challenges and limitations of NLG appear:

- **Data quality** - To generate the desired output, the data provided must have an almost perfect quality which many times are very challenging(e.g. without errors)
- **Originality** - Some tasks, like chat-bots, have a strong limitation of not knowing how to answer questions they have never seen (e.g. data specific to one domain).
- **Integrity of the data**: NLG systems are based on ingested data. Some of this data may contain serious mistakes (e.g. data not corresponding to standard ethics).

2.2 Computational Linguistics

The area which emphasises aspects of human language is Linguistics, and it concentrates on conducting systematic investigations into the features of both specific languages and of language in general. Computational Linguistics, on the other hand, uses the computer science field to provide methods for dealing with a large variety of natural languages (Grishman, 1986). When combining this field with the engineering side, it is possible to create intelligent systems like the ones presented in Natural Language Processing (Section 2.1). Both NLP and Computational Linguistics work interchangeably, where NLP tries to understand and process the natural language, while Computational Linguistics is more

concerned about the representation of computers and basic language as one. One main difference between them is in terms of tasks: Computational Linguistics takes the theories of theoretical linguists and tests them via an automatised process(e.g. testing different grammar concepts); NLP leads the process of automatic annotation of sentences where each word is assigned part of a speech tag (e.g. noun, verb, adjective).

Since the early years of research in this area, the attention was focused on developing three core applications (Grishman, 1986):

- **Information Retrieval (IR)** - It is the task of extracting any plausible and useful information from large collections of data, usually text (Manning, 2008) (e.g. Searching for specific names in huge raw text).
- **Machine Translation** - Its purpose is to use vast dictionaries and sets of linguistic rules to carry out the entire translation process from a source text to final output without the aid of a human (Slocum, 1984) (e.g. Translation of Chinese language to English).
- **Man-machine Interfaces** - Methods that can interact with the user using human language to reach the desired output(e.g. communicating with the database for data retrieval)

To achieve these types of systems, it is crucial to use the engineering side, as analysing and using syntactic, semantic and domain knowledge is NLP core job.

2.3 Mathematical representation of words

With recent advances in Deep Learning approaches for processing human language, dealing with huge quantities of it requires more effort to extract a sentence's semantic meaning than just linguistic analysis. For that purpose, word representations emerged: mathematical vectors where each dimension could represent a feature or a semantic meaning (Turian et al., 2010). There are many ways of creating such representations, from more naive approaches to Deep Learning ones.

One of the more classical approaches is Bag-of-Words (BOW) (Harris, 1954). This approach, represented in Figure 2.1, describes the occurrences of words inside a document. It takes a set of vocabulary and measures how many times each word takes place in a certain document. Although this approach is simple and offers flexibility for customisation on a specific set of documents, if a huge vocabulary is presented, the vector's length will also increase, creating a sparse matrix containing no order of words used in the sentences.

A good method to use in lexical analysis, and an improvement of BOW, is Term Frequency-Inverse Document Frequency (TF-IDF). It uses the concept of weighted word representations, which means instead of only counting the number of word occurrences in a document, the representation is created based on word frequency which would contain information about the most and less occurred words (Jones, 1972). TF-IDF of a word is calculated as

$$TF - IDF(t, d, D) = TF(t, d) * \log(\frac{D}{dft}) \tag{2.1}$$

Where t is a word, d is represented as a document of t, D is a collection of documents, and dft is the sum of documents d with the word t in it. Both BOW and TF-IDF methods do not represent the order of words, their semantics or syntactical information and suffer from the curse of dimensionality. These limitations were the base for creating more powerful and robust methods for word representations, including deep learning models used as feature learning methods. Naseem et al. (2021) defined word embedding as a feature learning approach where a word could be represented in a N-dimensional vector space. To create language models capable of generating those embeddings, several techniques emerge and are classified in:

- **Unsupervised Learning** - which discovers hidden information in the context without any labelled data, for example, model *word2vec* presented by Mikolov et al. (2013).
- **Supervised Learning** - where the model has access to labelled data and is trained to mimic the output from the training on unseen data. Melamud et al. (2016) has created a model which has used two billion words as a learning corpus.
- **Self-Supervised Learning** - here, the model would be trained on a corpus where the system automatically generates the labels. The most prominent example of this method is Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018)

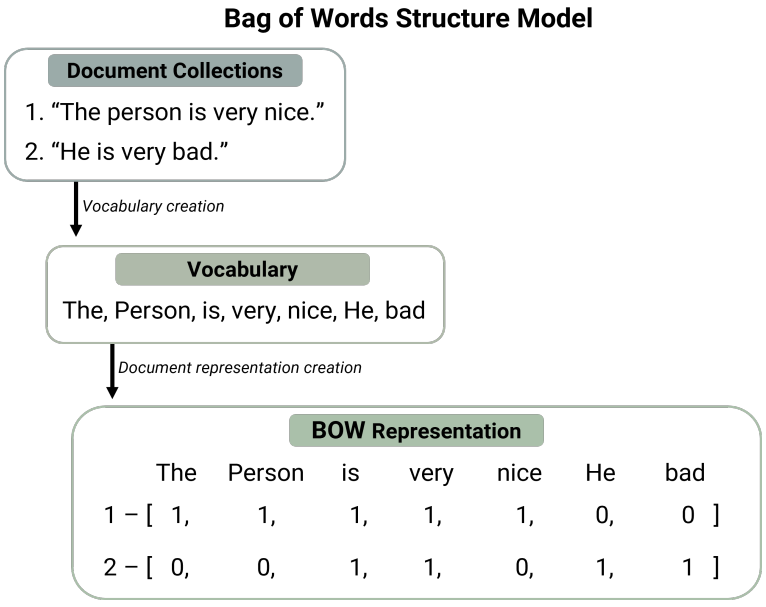

Figure 2.1: Overview of BOW model.

Training the models to represent words as vectors by these methods significantly enhanced the accuracy of the outcomes but presented limitations. For example, the vectors produced by *word2vec* model could not handle well Out-of-Vocabulary (OOV) words, assigning a random vector to every word that is not in the vocabulary.

2.4 Transfer Learning

For many language models, there is one limitation in common, when trained on a specific task, they can present problems in generalising to similar tasks. For instance, a language model with knowledge gained in recognising person names could be misled on a task to recognise car names. This problem led to the creation of pre-trained models using *Transfer Learning*, where a model is trained to represent a generalised type of knowledge and to be later fine-tuned on a specific one. This process does not necessarily require much data as the core knowledge was already taken. According to Pan and Yang (2009), *Transfer Learning* seeks to improve the learning of the target predictive function *ft* for the target task *Tt* in target domain *Dt* using the source domain *Ds* and source target *Ts* where the $Ds \neq Dt$ and $Ts \neq Tt$. Here the *Ds* is a domain pair represented by a feature space *X* and a marginal probability distribution $P(X)$. A task is a pair *T* with a label space *Y* and a predictive objective function which can be probabilistically represented as $P(y \mid x)$. Given that, it is possible to identify three main types of *Transfer Learning*:

- **Inductive Transfer Learning** - the ability to learn how to solve a specific task by having knowledge of a similar one, whether the target and source domain are the same.
- **Transductive Transfer Learning** - First presented by Arnold et al. (2007), where both the source and target tasks must be the same and could have different domains, and the target domain must be seen during the training.
- **Unsupervised Transfer Learning** - Similar to the first method, the model predicts the target label using the target and source domains without knowing the target labels.

This type of learning provided improvements in the NLP field, where users earned access to models of high quality and benefited from starting to solve problems from specific points instead from scratch. However, the tree types of *Transfer Learning* presented assume that the source and target domains have the same feature space, which limits their applications. To tackle this limitation, *Heterogeneous Transfer Learning* (Shi et al., 2010) was developed to resolve this issue and others like different data distributions. Furthermore, it allowed expanding the activities of *Transfer Learning* models into text-to-programming languages, cross-domain text categorisation and others.

2.5 Neural Search

The application of shallow and deep neural networks for the task of IR is named Neural Search (Mitra et al., 2018). These models aim to improve the precision and efficiency of search systems to understand and represent the meaning and context of queries and documents. Traditional IR systems mainly rely on keyword-search engines or statistical methods to calculate the relevance of a document to a query. However, these approaches do not catch the intent of a query to the documents, resulting in limited effectiveness.

Neural Search models are often trained on large amounts of labelled and unlabelled data to learn the generation of contextualised embeddings, which allows for capturing the

semantic relationship between words, sentences or documents. It is possible to fine-tune the model or use a pre-trained one on large corpora of text data to enable its generalisation capabilities. The process of Neural Search is divided into three main phases (figure 2.2):

- **Query Representation** - The first step is composed of encoding the desired query, which specifies all relevant information needed into a N-dimensional vector space.
- **Document representation** - The same process of encoding the query proceeds with all sets of documents to be searched, which captures the information contained in each document.
- **Relevance Calculation** - In the last step it is executed a relevance calculation (e.g. cosine similarity, query likelihood) between two representations to match the most relevant document to the query presented.

Using Neural Search models for IR greatly boosted the accuracy of retrieved documents than the keyword-search engines or statistical methods. Although the lack of high-quality data and processing power makes it difficult to train such powerful systems, sometimes constraining the domain to the training data.

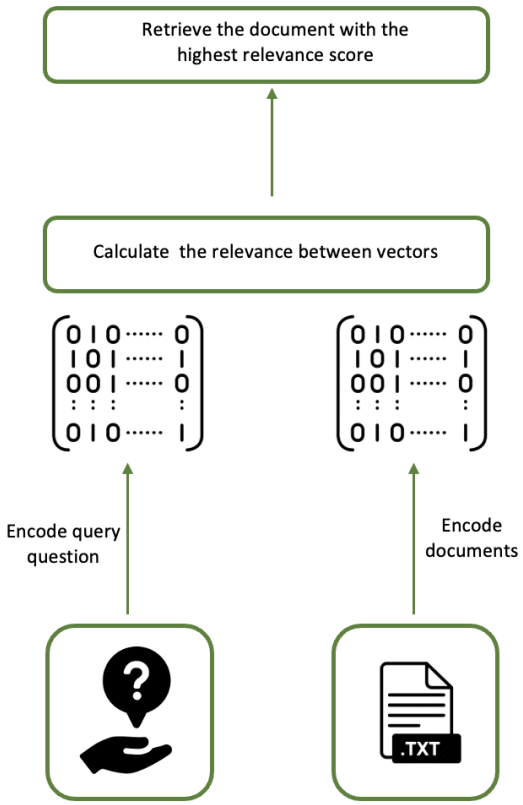

Figure 2.2: Overview of a Neural Search system

Chapter 3

Related Work

3.1 Natural Language Interface for Databases

NLI are systems created to make the interaction between humans and machines by natural language possible. The NLIDB has a more limited scope, they are designed to allow the interaction between users and databases. The user can provide any input, whether a voice command or a textual question, and the machine replies with an answer that can trigger an action or return data frames, text, or numbers. Its use appeared in the late '60s with the development of several well-built systems such as *LUNAR*, *SHRDLU* or *MARGIE* (Winograd, 1974). These systems had a more narrow window of a domain, restricting it to a specific one so the quality of the output could be more accurate. Currently, there are more intelligent systems capable of accepting questions from different domains such as *ALEXA*, *SIRI*, or *CORTANA*. All of those interfaces tackle the task of transforming any user request into machine-reproducible language and, from it executing an action. From this, it is possible to state that the most significant advantage of such interfaces is that users with low or null experience in computational resources can access and use them for their purposes. There are some limitations, such as the impossibility of the machine retrieving information or executing a task from an unrecognisable command. For example, in an interface that accesses information about patients, it is possible that a user can ask the system "*Why the patient has died?*". The meaning of this question can be simple, but naturally, the interface was not designed to answer those questions.

RDBMS (Codd, 1970) is a type of information storage system for maintaining relational databases that are still used today. The standard user for this type of system is non-technical with a strong knowledge of Management or Finance, so to access any information inside, he must have appropriate knowledge in SQL (Chamberlin and Boyce, 1974) given it is a core language of the system. Due to the shortage of a knowledgeable audience, the creation of NLIDB was crucial to tackle the problem of transforming natural language into SQL.

3.1.1 Classical Approaches

The task of transforming requests into SQL is a long research line of NLIDB that started mostly with the already mentioned *LUNAR* system. It was designed for research geologists to access the chemical analysis of lunar rock and soil derived from Apollo moon

missions (Woods, 1973). To create this system it was used a parsing method, but many others exist for NLIDB such as:

- **Keyword-Based** - The fundamental part of this method is to find keywords in the users' questions and match them against the database and its metadata. A clear example of this technique was introduced by Blunschi et al. (2012), where the *Search over Data Warehouse* system matches keywords against the database and produces a graph from metadata with all the nodes found, and later it assigns a score to each node to find the appropriate table/column from the database. This specific method is used in other systems such as *NLP-Reduce* (Kaufmann et al., 2007) or *QUEST* (Bergamaschi et al., 2013). Each of them has a different approach to the problem but still with the core component of matching tables, columns, or operations by keywords in common. It is a flexible and simplistic approach, but its main disadvantage is not being able to generate more complex queries, for example, queries with aggregations.

- **Pattern-based** - In this method, the system can use humanly created patterns and rules to compare them to an input sentence and verify if the pattern matches. It applies a predefined rule to formulate the query clause if there is a match. The purpose of this method is to use, for example, a word pattern *"by"* or *"per"* to understand that the output must be aggregated to something and if the system matches these keywords in the sentences, it adds a *Group By* clause. Figure 3.1 clarifies the vision described above. The patterns do not need to be precisely only keywords and can also be represented as a sequence of part of speech tags or other syntactic representations. An illustrative implementation of this method is the system *Savvy* (Johnson, 1984).

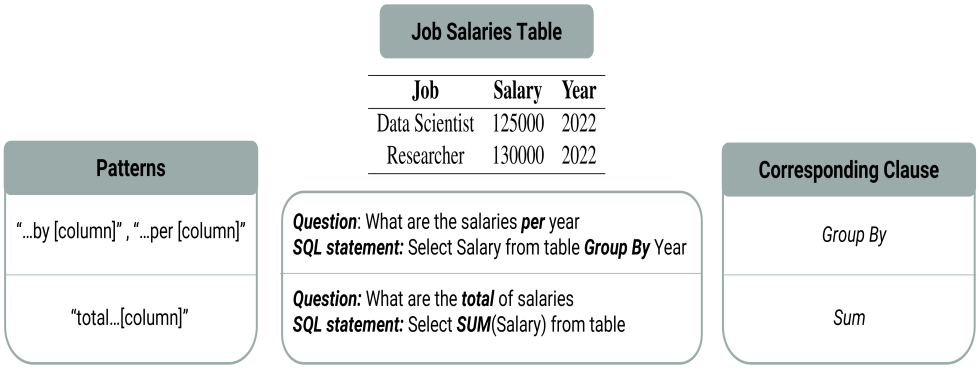

Figure 3.1: Overview of Pattern Based component.

- **Parsing-based** - This method is sub-divided into a syntactic or semantic-grammar one, as displayed in Figure 3.2. The syntactic method tries to generate a sentence's syntactic structure by breaking it into tokens and analysing the grammatical rules. This structure generates a parsing tree where some nodes are mapped into their semantic labels and then to the SQL. By analysing this structure, it is possible to visualise how column words can be related to aggregation words such as *"by"* or *"per"*. This method is used to implement the *NALIR* (Li and Jagadish, 2014)

system, where the tree nodes are mapped into SQL components rules to match inputted questions. The semantic-grammar parsing method is very similar to the syntactic one, but instead of having a full tree, it eliminates or groups together insignificant nodes, reducing the tree's complexity. Notably, the nodes are classified in terms of function or meaning rather than syntactic categories. It is also possible to assign a specific name to each node to reduce the ambiguity of inputted questions, but this requires prior knowledge of the elements in the domain. A system built using this method is *LADDER* (Hendrix et al., 1978).

- **Grammar-based** - This type of method uses a grammar set of handwritten rules to define questions that the system can answer. While typing the question, it is suggested to the user the next words for the correct input of the question, which in its way, reduces the ambiguity of it. A very big disadvantage of these systems is that they are highly dependent on the domain of written rules. An example of this method is a *TR Discover* (Song et al., 2015), which uses the auto-suggestion mode for the user input, so when the user writes *"p"*, it will accomplish the word by selecting *"profession"* and then try to predict the next word from the known grammar.

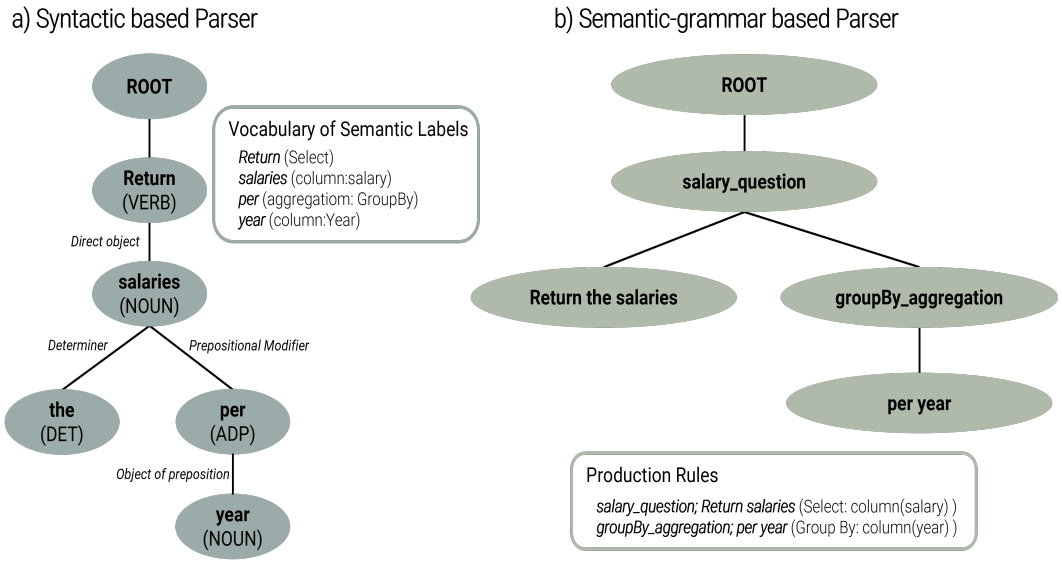

Figure 3.2: Two Parsing-based methods. a) shows how a syntactic tree is built based on its grammatical structure; b) represents a semantic-grammar tree where nodes match the production rules.

All of the aforementioned methods have been widely used over the last 5 decades, 3.1 demonstrates several NLIDB for SQL distinguished by the method used. Nowadays, the problem of text to SQL has earned new and more innovative approaches, such as using deep learning methods.

3.1.2 Deep Learning approaches

Deep Learning approaches have been used in various areas such as Computer Vision or NLP, yet, they require a large amount of data to rely on their training. Given the

Table 3.1: Classical NLIDB distinguished by method used to tackle the problem of text to SQL

Method	Model	Authors
Keyword	SODA	Blunschi et al. (2012)
	NLP-Reduce	Kaufmann et al. (2007)
	Précis	Simitsis et al. (2008)
	SQAK	Tata and Lohman (2008)
	Keyword++	Ganti et al. (2010)
	QUEST	Bergamaschi et al. (2013)
Pattern	Savvy	Johnson (1984)
Parsing	Lunar	Woods (1973)
	LADDER	Hendrix et al. (1978)
	WASP	Wong and Mooney (2006)
	USI Answers	Waltinger et al. (2013)
	NaLIR	Li and Jagadish (2014)
	ANTHENA	Saha et al. (2016)
	SQLizer	Yaghmazadeh et al. (2017)
	BioSmart	Jamil (2017)
Grammar	TR Discover	Song et al. (2015)

recent advancements in this field, complex datasets such as *Spider* (Yu et al., 2018b) and *WikiSQL* (Zhong et al., 2017) were created and the task of NLIDB has reached new levels of performance. To achieve this, several new deep learning approaches arose:

- **Sequence to Sequence** - One of the first approaches to appear was *"End-to-End"* or Sequence to Sequence (seq2seq) (Sutskever et al., 2014), where the model is trained on input-output pair of statements in the task of machine-translation. It uses a multi-layered Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), encoding the input sequence into a fixed dimensions vector and decoding it with the same technique into target sentences. Zhong et al. (2017) modified this to a text to SQL task, proposing an Encoder-Decoder system with an Attention layer and a reward mechanism in the query generation loop. Using this model, they also proposed a novel dataset *WikiSQL*, used as training pairs. However, the models trained with seq2seq approach have a problem of *"order matter"* where the training questions have a specific order to formulate SQL clauses. For example, the training question "*What are the salaries where jobs are Data Scientist and Research Scientist?*" would formulate the correct SQL clause. However, if the question's order is changed to "*Where jobs are Data Scientist and Research Scientist what are the salaries?*" the model presents difficulties given it has never seen the second question, although it has the same exact meaning. Another worthy of mention method is the database schema integration into the training phase. The model *X-SQL* (He et al., 2019) has included the encoded schema information with a sequence encoder to improve the decoding step with contextual information from the schema.

- **Sequence to Set** - To overcome the order problem, a new method was developed where instead of trying to predict the proposed SQL directly to its clause, the question is broken into parts. Each part is then assigned to a respective clause (i.e. Select, Where, Aggregation clauses). *SQLNet* (Xu et al., 2017) is proposed as a Sequence to Set (seq2set) approach. This model employs a sketch-based and slot-filling architecture, where it aligns slots naturally to the syntactical structure of SQL query. Later the neural network is trained with the attention mechanism to predict each slot. With the *SPIDER* dataset development, the schema of the database received increasing attention as the evaluation phase occurred in the complex and unseen database. Bogin et al. (2019) developed a Graph Neural Network (GNN) to encode the database schema to later feed this representation into an Encoder-Decoder architecture. However, using this method brings another issue, like context ignoring, as it is possible to lack information for a specific prediction.

Both the methods mentioned above have been vastly explored. Even though they present some hurdles, the researchers tried to get around them by implementing more complex structures into existing models. For example, the model *RASAT* (Qi et al., 2022) integrated with a seq2seq architecture from the *T5* model (Raffel et al., 2020) swapped the self-attention layers in the encoding phase by relation aware ones, resulting in two additional relation embedding lookup tables. Consequently, it converts the input sentences into an interaction graph by adding the relations through relational propagation. Finally, when the model is trained, it retrieves the relation embeddings from embedded lookup tables through an interaction graph. This results in a model with almost all literature relations, schema linking system and syntactic dependency embedded into one relation representation. Many other models were integrated as the ones that receive feedback from the user to verify generated queries (Elgohary et al., 2020). The model generates an answer, and if it is incorrect, the user can provide additional input to reformulate the query. From the current situation, it is impossible to confirm that the text to SQL task is completed and there is nothing to improve as none of the models have achieved a 100% accuracy on either *SPIDER* or *WikiSQL*. Table 3.2 highlights most of the models used in this task.

3.2 Word Embeddings

Several language models have been developed in order to extract the meaning of user input because human language can be ambiguous or vague, which also permitted the evolution of NLIDB. The models are trained with different techniques, as mentioned in Section 2.3, to produce embedded vectors of N dimensional space to learn features from words or sentences. Many approaches create word embeddings from a text corpus, and it is crucial to refer to them as they are one core component in this work. The need for this component exists as it is possible to calculate the similarity of a given pair of words or sentences considering their context and meaning. For instance, the words *"jobs"* and *"professions"* are more similar to each other than to two other words *"year"* or *"date"*, and this similarity is necessary to map the SQL clauses correctly. Figure 3.3 is a simplified example in three-dimensional space for better visualisation.

Table 3.2: Deep Learning models employed to build NLIDB, using the WikiSQL and Spider datasets.

	Model	Architecture	Authors	Dev	Test
WikiSQL	Seq2SQL	Seq2Seq	Zhong et al. (2017)	60.8	59.4
	PT-MAML	Seq2Seq	Huang et al. (2018)	68.3	68.0
	SQLNet	Seq2Seq	Xu et al. (2017)	69.8	68.0
	TypeSQL	Seq2Set	Yu et al. (2018a)	74.5	73.5
	X-SQL	Seq2Seq	He et al. (2019)	89.5	88.7
	HydraNet	Seq2Set	Lyu et al. (2020)	89.1	89.2
	SeaD	Seq2Seq	Xuan et al. (2021)	90.2	90.1
	SeaD + Execution Guiding Decoding	Seq2Seq	Xuan et al. (2021)	92.9	93.0
Spider	GrammarSQL	Seq2Set	Lin et al. (2019)	34.8	33.8
	GNN	Seq2Set	Bogin et al. (2019)	40.7	39.4
	IRNet	Seq2Set	Guo et al. (2019)	53.2	46.7
	RATSQL	Seq2Seq	Wang et al. (2019)	62.7	57.2
	Photon	Seq2Seq	Zeng et al. (2020)	63.2	-
	ValueNet	Seq2Set	Brunner and Stockinger (2021)	-	62.0
	RASAT + PICARD	Seq2Seq	Qi et al. (2022)	75.3	70.9
	T5-3B+PICARD	Seq2Seq	Scholak et al. (2021)	75.5	71.9
	LGESQL + ELECTRA	Seq2Seq	Cao et al. (2021)	75.1	72.0
	RESDSQL-3B + NatSQL	Seq2Set	Li et al. (2023a)	80.5	72.0
	S^2SQL + ELECTRA	Seq2Seq	Hui et al. (2022)	76.4	72.1
	N-best List Rerankers + PICARD	Seq2Set	Zeng et al. (2023)	76.4	72.2
	SHIP + PICARD	Seq2Seq	Zhao et al. (2022)	77.2	73.1
	Graphix-3B + PICARD	Seq2Seq	Li et al. (2023b)	77.1	74.0

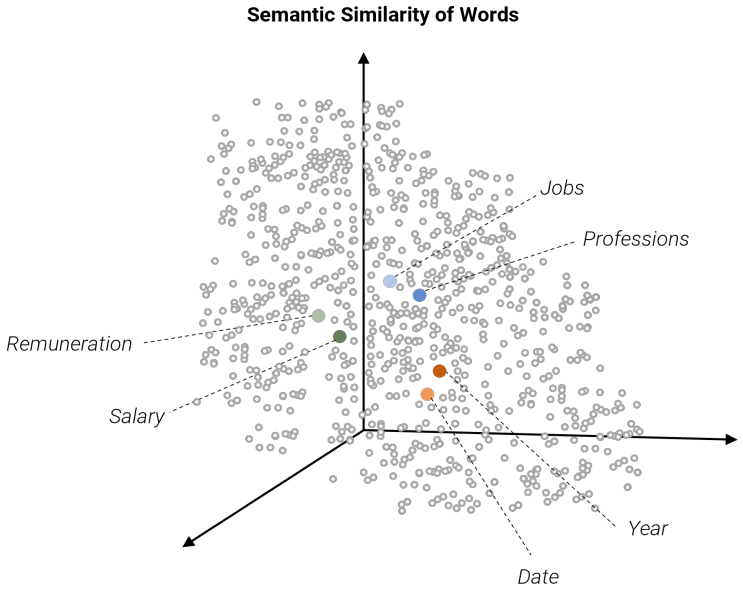

Figure 3.3: Semantic similarity of words in a three-dimensional vector space.

3.2.1 Word2Vec

The *Word2Vec* model made a breakthrough in word representations using Unsupervised Learning (Mikolov et al., 2013). The architecture of this model allowed it to either use a *Skip-Gram* or Common Bag of Words (CBOW) model based on a feed-forward Neural Network. Using *Skip-Gram*, *Word2Vec* is trained by trying to predict the context words from its target word. The CBOW tries to predict the target word from the vector representations of words in its surrounding. The main goal of *Word2Vec* is to learn the semantic meaning between words using their context. For example, the words *"jobs"* and *"professions"* would be represented close to each other due to their semantic meaning. This model brought three main benefits. When trained with *Skip-Gram*, the model can capture the OOV words due to its context concentration. In another way, if the model is trained using CBOW the training phase would be much faster given that CBOW sees the whole context of a sentence as one. Also, a great advantage was that the vectors produced by *word2vec* are low dimensional while still preserving the semantic meaning in each dimension, unlike BOW or TF-IDF. Even with the improved quality to tackle OOV words, still struggles to achieve significant results, being its main limitation.

3.2.2 Pre-trained Language Models

The production of word embeddings has its own drawbacks as the OOV words, or limited semantic context is restricted to the availability of the training datasets. The Pre-Trained Language Models (PLMs) have earned their popularity because of the ability to transfer already known knowledge to other specific tasks, being very successful and reaching state-of-the-art performances. BERT is one PLMs, and it is trained using the idea of Self-Supervising Learning. It leverages the Transformer architecture and is trained only in large text corpus while considering the context of right and left tokens at each step. This model is pre-trained using two functions. The first is the masked language modelling that hides 15% of the input and tries to predict it. The second function is Next Sentence Prediction which tries to predict two sentences by predicting the second and verifies if it follows the same order as the training corpora. The main benefit of BERT is that it has the ability to process a large amount of text and increase its performance over legacy methods. With this amount of knowledge that the model poses, it has the drawback of being very time-consuming. If the user wants to retrain the model in other non-related tasks, given that all the weights must be updated, it takes considerable time.

Other models as Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu et al., 2019) and A Little BERT (ALBERT) (Lan et al., 2019) were developed to improve the original BERT model. While the RoBERTa model adds more training data and parameters, the ALBERT reduces the parameters and proposes new architecture. Given that by increasing parameters, a model can present improvement in performance, an innovative approach was proposed using the auto-regressive model called *GPT-3*. This model is trained with 3 trillion words incorporating data from several existing databases such as *Common Crawl*, *Wikipedia*, and many others. It has 175 billion parameters and has shown a successful performance on several tasks such as *Question-Answer*, *Translation* and others. Even with the increased performance and better models each time, challenges and limitations still exist. The biggest to consider is the environmental impact, as the more parameters added to the model, the more training time it takes. With its increase,

it raises energy consumption, so in terms of future work, it is crucial to review how the training time can be reduced.

Chapter 4

Methodology

4.1 Overview

This section presents the methodology to build and evaluate a NLIDB system, the product of this project. The methodology comprises four main phases. The first phase consists of NLIDB preparation, where a set of pre-defined activities is passed through in order to arrange the environment for a text-to-SQL task. Subsequently, there is the translation phase, which is dedicated to present the core of this work to tackle the text-to-SQL task. The third phase performs the execution of the generated SQL code. Finally, the last phase is dedicated to execute the evaluation process of the algorithm's performance against a chosen benchmark. Figure 4.1 illustrates the four phases in detail.

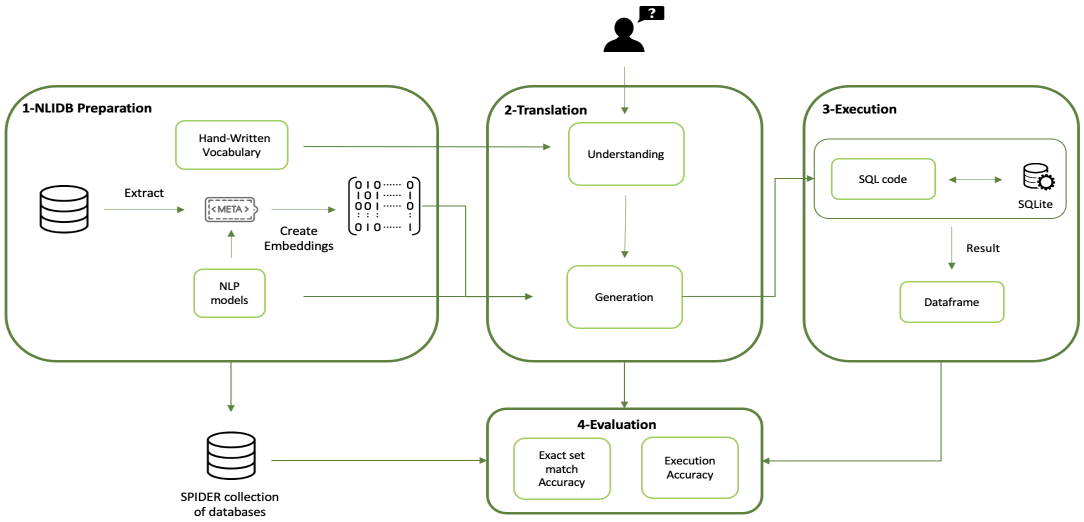

Figure 4.1: Overview of the methodology

4.2 NLIDB Preparation

To translate a natural language text to SQL code, it is necessary to understand the meaning of a question in natural language which can lead us to several potential issues, such as

ambiguities. Consequently, this phase is focused on executing several tasks to reduce ambiguities and recognise entities embedded in the data.

The first step is to choose the pre-trained NLP models which were adopted to create mathematical multidimensional representations (*word/sentence embeddings*). Four models were chosen, which are hosted publicly in the *HuggingFace*[1] platform. Table 4.1 represents each with their performance on the semantic search task, the base models used for fine-tuning, and the number of dimensions in the produced vectors.

Table 4.1: Fine-tuned models for semantic search and related tasks and their performance

Fine Tuned Model	Base Model	Performance	Nº Dimensions
multi-qa-mpnet-base-dot-v1	*MPNet* (Song et al., 2020)	57,60	768
all-mpnet-base-v2	*MPNet* (Song et al., 2020)	57,02	768
multi-qa-distilbert-cos-v1	*DistilBERT* (Sanh et al., 2019)	52,83	768
multi-qa-MiniLM-L6-cos-v1	*MiniLM* (Wang et al., 2020)	51,83	384

The second step in this phase is to extract the characteristics and elements in the data in order to recognise the entities of the data used in the experiment. For example, given the database, every table and column name are embeded into an N-dimensional vector with the intent to compare the similarity between sentences/words.

Many SQL functions represent single keywords. For example: ”mean” is referred to *AVG()* SQL function, and the word ”where” or ”whose” could be interpreted as a *WHERE* clause in most of the questions. Given this scenario, this section also has an extensive vocabulary with synonyms and keywords of all possible SQL functions that can appear in questions. Both the metadata embeddings of a database and the vocabulary are stored and then implemented into the Translation step, where the algorithm generates the SQL query from the question.

4.3 Text to SQL Translation

This section describes how to translate natural language sentences into SQL code. The proposed method in this work produces SQL queries that combine pattern and syntactic based approaches. While these approaches, separately, create only domain-specific patterns of keywords or parsed trees to produce the target query. This method focuses on building the query from specific linguistic patterns generated by syntactic trees from the question interpretation. With the objective to generate a valid query code the following steps are necessary:

- **Interpret the question** - The first step is to transform the given natural language question into a valid form to be inserted into the model. The question is processed using *Stanford Core NLP* (Manning et al., 2014) pipeline and extracted the Part-of-Speech (POS) tags and the dependencies between every word that is considered a ”*Noun*”, ”*Proper Noun*” or a *”Verb”*. Everything that was extracted is stored in an ordered way and is introduced to the lexical pattern matcher.

[1] https://huggingface.co/models

- **Match the correct patterns** - The stored sequence of words is now matched into specific hand-written lexical and syntactic patterns. For example, the sequence "*head names*" in the question "*What is the head names?*" corresponds to a two-word pattern where the word "*names*" is a noun and also a *compound* of the word "*head*". The *compound* dependency represents the word that is syntactically related to another, which could be also a verb or numbers. Those two words can be coupled together as they represent the semantic meaning of an *entity* that can be understood as only one data concept.
- **Predict the columns** - With all patterns found, they are evaluated by using the same models from section 4.2. Those models generate pattern embeddings which are used to predict the correct table and column names.
- **Formulate the query** - The last step uses the previously created vocabulary in the section 4.2 to select if any of the columns correspond to SQL functions and formulates the query. Therefore, after the mapping step is completed, the algorithm has the conditions to assemble the query with all necessary "*JOIN*" functions. For example in the question "*What is the average age of heads*" the algorithm selects the column *age* from the table *head* and adds the average function to it.

The formed architecture is also considered a *slot-filling* approach as it matches everything to the respective clauses. Hence, the input for this section is the question, created vocabulary, database, data embeddings, and NLP models which produce a SQL code. Figure 4.2 demonstrates how each step of this section is formulated and how the final query is mapped out.

4.4 Execution

As mentioned in the previous phase, the output of the translation phase is a SQL code. Therefore, to produce the results intended by the question, it must be executed in a database engine capable of running SQL query. By directly communicating with *SQLite*[2] the query is executed with a specification of which database to iterate. In this work, it is adopted the aforementioned database engine, as the benchmark dataset which was chosen to evaluate, consists of multiple databases stored with the exact same engine. For a better understanding of the outputted data, it is prepared a new view of it as the columns displayed have no order and in some cases can confuse the user. The columns, which are the direct output of a SQL functions as well as aggregated columns are displayed at the end of the dataframe.

4.5 Evaluation and Experiment

To test the text-to-SQL performance, it is necessary to choose the correct data. The *SPIDER* (Yu et al., 2018b) dataset is a complex benchmark where the most prominent semantic parsers are tested in the text to SQL task at the moment. This dataset comprises

[2]An SQL database engine that is quick, self-contained, highly reliable, and the most used database engine worldwide https://www.sqlite.org/index.html

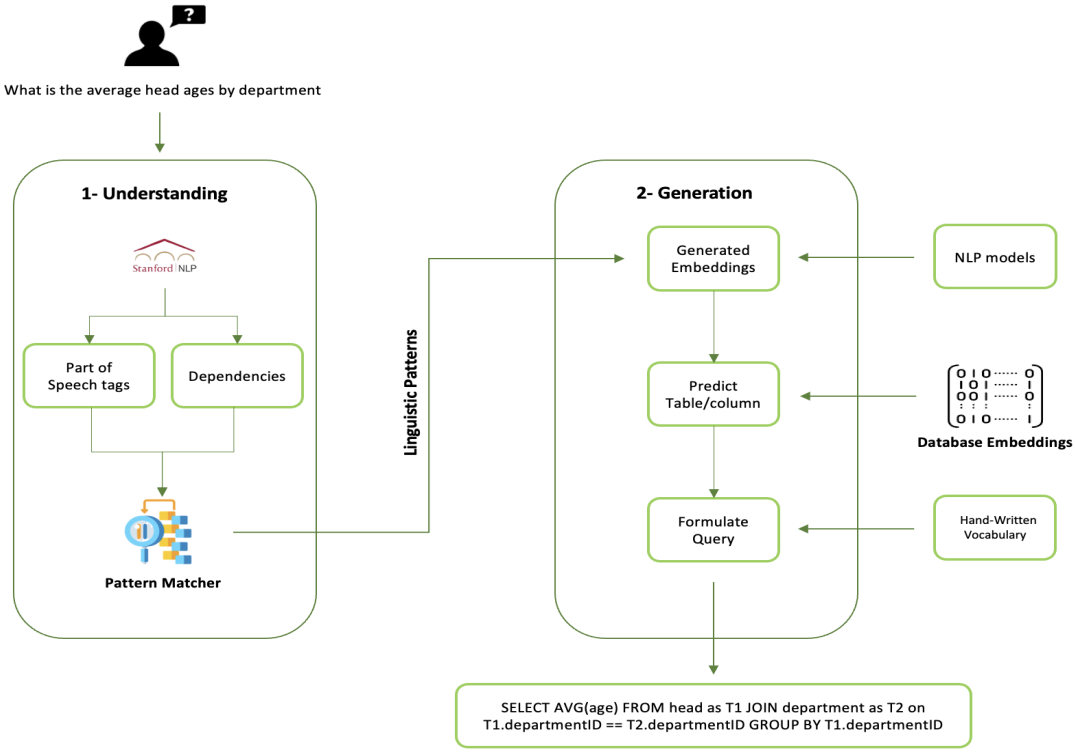

Figure 4.2: Understanding step illustrates the walk-through of the inserted question as well as the extracted items. The Generation shows the prediction and formulation steps

200 databases with multiple tables and 10.181 questions corresponding to 5.639 complex and unique SQL queries. The decision to rely on *SPIDER* data comes from it being cross-domain and having a very diversified type of queries covering the majority of SQL components (e.g., "GROUP BY", "ORDER BY", "SUB-SELECT", "HAVING"). However, many databases from this benchmark do not follow the general RDBMS naming conventions (Papamichail et al., 2020). For example, columns with names such as "*f1*" and "*f2*". Rather than focusing on the whole benchmark, specific databases were chosen. Where the primary and foreign keys are clearly identified, and all columns are interpretable. It is also excluded from the evaluation the questions which require *EXCEPT*, *INTERSECT* and *UNION* SQL clauses because the main goal is to prove the generalisation capabilities of the algorithm and the capacity to structure a working query. The entire benchmark is also executed using two embedding models to compare the execution accuracy and computational time with two distinct similarity metrics.

The pre-selected databases from the train and development set of the benchmark can be visualised in Table 4.2, along with the total number of tables and columns from the databases within each set. The train and development sets consist of 429 and 120 questions, respectively (the questions are identified in the appendix A). The questions from the train set were used to improve the linguistic patterns and the rules employed for query structuring. The goal was to improve the algorithms' ability to interpret natural language questions accurately. On the other hand, the databases from the development set were used for testing purposes, evaluating the algorithm's effectiveness in generating working SQL queries. Overall, the selected sets provide a comprehensive evaluation structure to

assess the proposed algorithm's cross-domain adaptability and capacity to structure working queries.

Table 4.2: Pre-selected databases from two sets and the total number of tables and columns in each set

Set	Databases	Total Tables	Total Columns
Train	book 2, gymnast, musical, farm, device, race_track, ship_1, school_bus, wrestler, department_management, climbing, journal_committee, bodybuilder, performance_attendance, election_representative	37	191
Dev	poker_player, singer, orchestra, voter_1	11	54

The produced queries are executed accordingly to the execution section 4.5 and evaluated with the author's dataset metrics. *SPIDER* authors propose two similar ways to evaluate a semantic parser, the exact set match and the execution with values. The first one evaluates the formulated query by its components: "SELECT", "GROUP BY", "WHERE", "ORDER BY", and all the SQL keywords which are not columns or operators and compares it to the gold query. The formulated query is only correct if all the components are correctly predicted. The second, execution with values uses a formulated output of the gold queries and provides execution accuracy. Finally, is conducted an evaluation report given the exact set match divided by several types of queries and execution accuracy of our model, to find where the proposed algorithm performed well or poorly.

Chapter 5

Text-to-SQL Algorithm

This chapter introduces two approaches to the text-to-SQL algorithm. The first approach focuses on validating the idea of incorporating linguistic patterns, pre-trained embedding models, and a hand-written vocabulary. The first section overviews the algorithm's general structure and presents its pseudo-code. The second approach outlines the algorithm designed for databases, highlighting the modifications made to the previous approach. The second section also presents the structure and pseudo-code of the actual algorithm, providing an overview of how it works.

5.1 Algorithm for Single Tables

Neural Search is one of the concepts covered in this work, where particular linguistic structures correspond to the database representation and, through the use of a handwritten vocabulary and structuring rules, reproduce a SQL query. Before introducing the main algorithm for the text to SQL conversion in database systems, it is worth recalling how the initial proposal was validated on single tables. When researching the possible ways of replicating a SQL code from natural language, a recurrent pattern was observed whenever displaying the final output. Specifically, the names of columns in any table were either a *Noun* or a *Proper Noun*. The Usta et al. (2021) also pointed to this finding, referring that *Noun* words are matched to the table or column names.

The pseudo-code nº 1 demonstrates the steps and iteration processes to formulate a SQL code from text. The process involves initialising two lists, *"final"* and *"POSList"*, which will be used to store the formulated query and the words in the question along with their corresponding POS tags, respectively. The variable *"pms"* denotes the pattern's maximum size, which means the number of words that can be used to form a pattern. Step 1 uses the tokenizer from *Stanford Core NLP* to break the question into tokens and iterate over them, extracting the corresponding POS. The next step is considered the most important as the function $extract\_column$ leverages the use of pre-trained models to embed and compare similarities of the discovered patterns within the columns. A short example is demonstrated in figure 5.1. Step 3 consists in tracking down the SQL functions by specific keyword patterns retrieved from the vocabulary. The latest 2 steps are designed to run the query and store the output in a JavaScript Object Notation (json) format.

Algorithm 1: Text to SQL for single tables

Input : Question, Column Embeddings, and Vocabulary
Output: Formulated SQL query and Json file

```
pms ← 3;
final ← [];
POSList ← [];
// Step 1: Extract Part of Speech (POS) tags
for token in tokenize(question) do
    extractPOS(token);
    Save the extracted POS with their tokens into POSList;
end
// Step 2: Find patterns and extract columns
for token in POSList do
    for s in range(0,pms) do
        final ← extract_column(token,s);
    end
end
// Step 3: Find SQL functions and build query
for token in POSList do
    aggregationsFunction(token, POSList, final);
    basicSQLfunctions(token, POSList, final);
    havingFunction(token, POSList, final);
end
// Add Limit to the end query
final ← final + "Limit 5";
assemble_query(final, POSList);
// Step 4: Execute query
result ← exec_query(sql);
// Step 5: Store the output in a JSON format
output ← json_output(sql, POSList, result);
```

The algorithm above was tested using a flat file in the comma-separated values (csv) format containing diverse information types. Based on the nature of the majority of questions that can be asked to a table, they usually involve references to one or more column names. Even if the names of the columns are altered to related words, they remain the same in terms of linguistic form. Given that, it was pertinent to compare the similarity between column names and words that pursue as a linguistic form *Noun* or *Proper Noun* to identify the related columns in the question correctly. Also, it was contemplated that if the word is followed by another one or two words representing a *Noun* or *Proper Noun* as a linguistic form, they are truncated together. This is because many names of the respective columns are represented as multiple words.

In some cases, as there is no explicit relation between words as dependencies or constituencies (Collins, 1997), there could be a mismatch of what is necessary to embed. This is because a single question can contain words as *Nouns* that do not represent a column

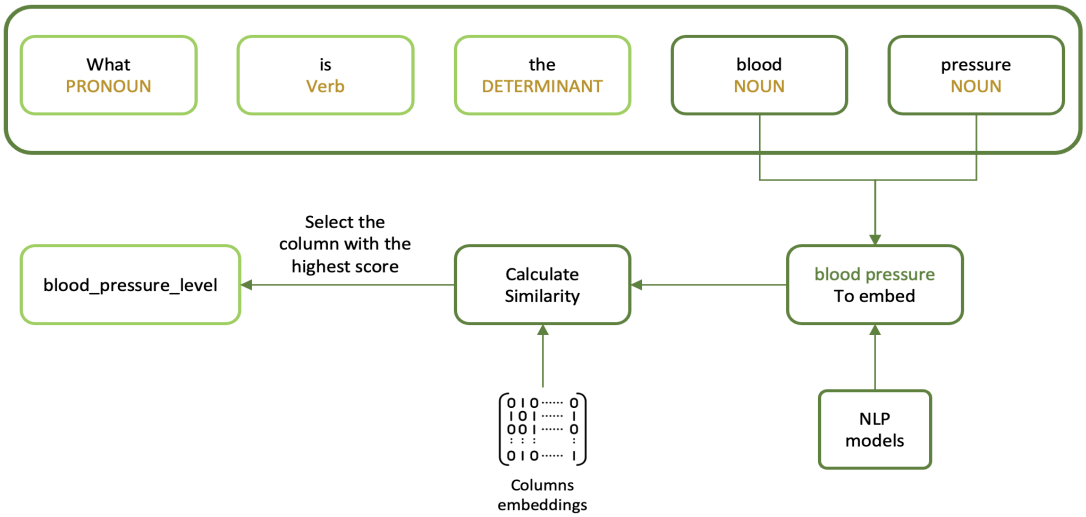

Figure 5.1: Example of extract_column function, where the function receives each token from the question and iterates only over these which are considered as *Nouns*. It recognises the pattern of two consecutive *Nouns* and saves them together. Lastly, the pre-trained model embeds these words as a whole sentence and calculates the similarity within the column embeddings. In the end, the function extracts the column with the highest similarity score in the search space

form from the table. To address this issue, a margin score is considered when calculating similarity. When a word is considered of a correct lexical form, the similarity between it and the columns embeddings must be superior to 0,35. For this purpose, cosine similarity (Salton and Buckley, 1988) is the preferred metric, given that it measures the orientation of embedded vectors rather than their magnitude and enables the capture of meaning between two sentences.

The mathematical formula for this metric is represented as:

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{t} \cdot \mathbf{e}}{\|\mathbf{t}\| \cdot \|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \tag{5.1}$$

Where $\mathbf{t}$ and $\mathbf{e}$ are two non-zero vectors with the same number of dimensions and $|\mathbf{t}|$ and $|\mathbf{e}|$ are lengths of the two vectors.

If the following word possesses a similarity greater than 0.35, it is considered when formulating a sub-query. After all the columns are discovered from the inputted question, a sub-query is formulated with a simple yet effective vocabulary. This matches every word within a set of pre-defined rules whose meaning could generate a SQL function in the context of a question. Finally, the last two modules would add the *LIMIT* function with a fixed amount of 5 lines into the query to save computational resources and store the output in the json format.

Overall, the steps that were made in this algorithm allowed the identification of columns of a table, as well as the SQL clauses, and correct structuring of the SQL query.

5.2 Algorithm for Databases

After the validation of the proposed methodology on tables, the next step involves adapting the algorithm to work with databases. However, there are notable differences in the set of questions that can be asked to a database, such as the possibility that a question may reference the table's name without requiring it to be explicitly selected. For example, a database with two tables named "*heads*" and "*departments*" and a request "*Show me the heads of the departments*". The aforementioned question references two tables, although when structuring a SQL query, it is necessary to know only the names of heads without any further concatenation. These type of questions brings a new level of difficulty to the older algorithm, as there are more than just columns to match, and it is mandatory to understand what to select and from which table. Given that, developing new linguistic and pattern rules and a new module to enable the concatenation of multiple tables is compulsory.

The following pseudo-code replicates all the necessary steps of the new algorithm for databases. As it is possible to visualise, the algorithm initialises two empty lists: one is "*final*" where the SQL query would be stored until execution and a "*pseudoQuery*" which stores the output of the preliminary functions. The first step involves extracting the POS tags and truncating the tokens corresponding to predefined linguistic rules. The following step iterates over the saved tokens and executes two functions that extract the correct table from a specific pattern and then predict its specific column. In step 3, the major difference is the linguistic rules associated with each of the functions that map SQL functions. Also, the inclusion of a function that executes the concatenation of tables if necessary. The last two steps contemplate the assembling of the final SQL query, its execution, and storing of the output. The storing function differs from the latest version as it is already prepared to retrieve the evaluation of the benchmark dataset's text-to-SQL task.

Algorithm 2: Text to SQL for databases

Input : Question, Database Metainformation Embeddings, Vocabulary
Output: Formulated SQL query, SQL output dataframe

$final \leftarrow []$;
$pseudoQuery \leftarrow []$;

```
// Step 1: Extract POS tags and truncate dependencies
```

for *token* ***in*** *tokenize(question)* **do**
  $pseudoQuery$.append(extractPOS(token));
end
$pseudoQuery \leftarrow$ truncateDependencies($pseudoQuery$);

```
// Step 2: Given the linguistic rules embed the
   patterns and extract table/columns
```

for *token* ***in*** $pseudoQuery$ **do**
  if *token* $\in$ *predefinedRules* **then**
    $tableColumn \leftarrow$ extract_table(token);
    $final \leftarrow$ extract_column($tableColumn$);
  end
end

```
// Step 3: Find SQL functions and structure the Join
   if necessary
```

for *token* ***in*** $pseudoQuery$ **do**
  aggregationsFunction(token, $pseudoQuery$, $final$);
  basicSQLfunctions(token, $pseudoQuery$, $final$);
  havingFunctions(token, $pseudoQuery$, $final$);
end
$final \leftarrow$ structure_join(pseudoQuery);

```
// Step 4: Assemble the query and execute it
```

$final \leftarrow$ assemble_query($final$);
$result \leftarrow$ execute_query($final$);

```
// Step 5: Save the data frame or the resulted value
   in a json format
```

$output \leftarrow$ json_output($final$, $result$);

Besides the differences identified in the pseudo-code, the two algorithms differ in four significant ways:

- Entities and attributes extraction - Data stored in RDBMS are consistent from different tables. To achieve its correct selection, the algorithm must possess a comprehensive understanding of the entities within the database, such as tables or columns, and the attributes that correspond to the values of each column. To allow the improvement of NLU capabilities, a new class of functions has been created that expands upon hand-written linguistic rules through collaboration with extraction functions.

- Improved Vocabulary module - To tackle many challenges created by user questions, special word patterns were added in combination with pre-defined linguistic rules to the already existent vocabulary. This allows an improvement in replicating

a SQL query. For example, the question "Show me the heads of departments" includes two tables "*heads*" and *departments*. The improved module understands that it is only necessary to select one column consisting of the table *heads* and column *names*.

- Additional module for *Join's* - In difference from the tables, extracting the necessary information from the databases may require concatenation of tables. To allow this to happen, a new module was created that examined every column in the pseudo-query and structured the *Join* clause. It follows specific rules which identify if there is more than one table referenced in the question and stores this information. Consequently, it retrieves all the identificational columns from the desired tables and embeds them into N-dimensional vector. Then it iteratively selects the best match of columns corresponding to the highest similarity. Lastly, it retrieves only two identificational columns which have the strongest match. This process is equally the same for two or more tables identified. Figure 5.2 depicts this process.
- Dependencies between words: Instead of calculating similarities for words in a range of three consecutive *Nouns* or *Proper Nouns*, which created many constraints in the step of extracting the table and the column name. The truncation of words was adopted for specific patterns as *compound* or *adjectival modifiers*, where both serve to modify the root of a dependent *Noun* or *Verb* in more specific cases. This allowed the algorithm to surpass the problem of a blind pattern prediction, as it incorporated words that do not represent any column or value from the table, lowering the similarity with the correct table/column.

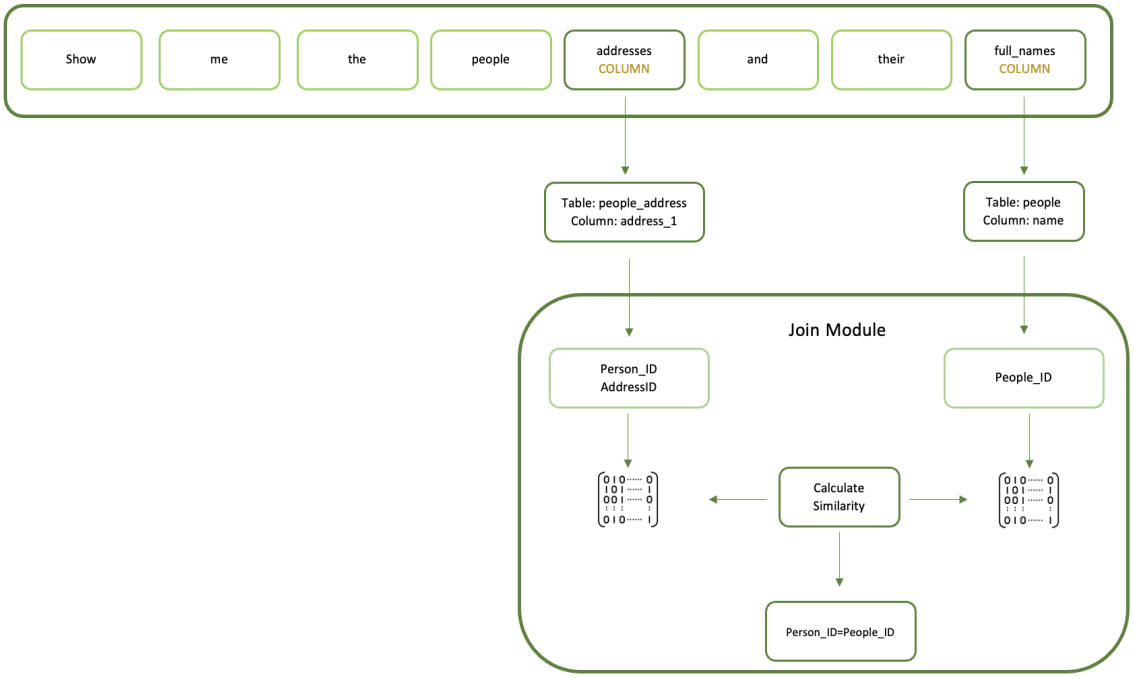

Figure 5.2: Overview of Join module steps

The revised algorithm for databases introduces some significant differences compared to the previous version. In addition to these significant changes, minor modifications were

made to the initialisation step of the table and column name embeddings. Specifically, in the previous algorithm, every column name was considered for embeddings, including those that do not carry any semantic meaning of a table, such as identificational columns like ”city_ID.” To address this issue, such columns are now excluded from the similarity calculation step. Furthermore, when a table contains a column named ”*Name*”, the algorithm adds a conjunction of words between the table and the column name (e.g., ”department_name” if the table is named ”department”) to the embedding list. These minor changes allow the algorithm to correctly identify and extract the semantic meaning of most questions.

The similarity algorithms used in calculating words and sentences, as discussed in 4.1, accept two similarity metrics: the dot product (Axler, 1997) and the previously mentioned cosine similarity. Despite their varying characteristics, the database algorithm implementation utilises both metrics to match the correct tables and columns. The performance of these metrics regarding accuracy in selecting the correct table and column is presented in the subsequent chapter, and the optimal choice is elaborated upon. The next mathematical formula denotes the dot product similarity metric:

$$\mathrm{dot}(u, v) = u \cdot v = \|u\|\|v\| \cos(\theta) \tag{5.2}$$

The u and v are two non-zero vectors the $\|u\|$ and $\|v\|$ are the lengths of the respective vectors, and θ is the angle between the two vectors. The algorithm is constituent of 5 main steps, as in the old version. However, only the mapping of SQL functions and the execution steps remained almost the same.

Chapter 6

Analysis of the Results

This chapter discusses several details and elements that impact the performance of the text-to-SQL algorithm proposed in Chapter 5, such as:

- Which pre-trained models from Table 4.1 provide better performance on the text-to-SQL task in the proposed algorithm?
- Which similarity metric identified in Chapter 5 has a better impact on the performance of the pre-trained models?

Firstly are presented the pre-selected databases, their composition and the results generated by the proposed algorithm using different pre-trained models and similarity metrics. Subsequently, the execution accuracy results are shown for the whole benchmark, and the algorithm's performance is discussed.

6.1 Results on the pre-selected databases

6.1.1 Execution Accuracy

Table 6.1 presents the execution accuracy using different pre-trained models for semantic search and performance using cosine and dot product similarity on both sets train and development. It is possible to visualise that the two pre-trained models which produce the best results are *multi-qa-mpnet-base-dot-v1* and *multi-qa-MiniLm-L6-cos-v1* where the first one scored bigger performance on train set and the last one on the development set.

Table 6.1: Execution accuracy of the proposed algorithm using different sentence embeddings on both sets with distinct similarity metrics. In bold are presented the two best results

Pre-Trained Models	**Cosine Similarity**		**Dot product Similarity**	
	Train	Dev	Train	Dev
multi-qa-mpnet-base-dot-v1	**72,03**	**70,0**	**72,03**	**70,0**
all-mpnet-base-v2	62,94	59,16	62,47	60,0
multi-qa-distilbert-cos-v1	68,53	64,16	68,06	65,0
multi-qa-MiniLM-L6-cos-v1	**69,46**	**70,83**	**69,46**	**70,83**

Taking a closer look at the performance, the model with the worst execution accuracy was *all-mpnet-base-v2* This is an unexpected outcome as this model is the second-best performer on the semantic search task from *sentence-transformers*[1] leaderboard. The reason is that all the other models were trained to be used specifically on semantic search tasks only, while this specific model was trained on general-purpose tasks varying between semantic search and sentence embeddings.

Regarding similarity metrics, both models produce similar performances. This occurrence happens because, to use the dot product metric, it is necessary to normalise the embeddings created by the models as our text-to-SQL algorithm uses thresholds. The magnitude on which the dot product differentiates from cosine similarity is removed when normalising the dense vectors. However, the models can reproduce different results depending on which metric the model was initially trained. For example, given the database "*farm*" with tables "*farm, city, farm competition, competition record*". The model *multi-qa-mpnet-base-dot-v1* using dot product metric finds the word "*competition*" more similar to the terms *farm competition*, which is the correct finding. In contrast, the other models relate it more to the words "*competition record*".

6.1.2 Time Execution

The model size was also considered to find the most suitable model for this task. Of the two best models, the *multi-qa-mpnet-base-dot-v1* is the biggest totalising 420 megabytes in size and producing 768-dimensional vectors, while the *multi-qa-MiniLM-L6-cos-v1* is only 80 megabytes in size and creates 384-dimensional vectors. The biggest model in size could be better to work on a broader scale where rare database domains appear as it has more potential to find related words. However, in this case, the smaller model is preferable as it is cheaper in computational power and faster to use in terms of time. The biggest model takes around 17 minutes and 39 seconds to run the whole training phase on average; it is 4.1 seconds for each question, while the smallest model takes 15 minutes and 40 seconds and 3.58 seconds on average using the cosine similarity metric. When changing to the dot product metric, it is possible to observe a decrease in the time taken to run the training phase. The biggest model has a slight decrease to 17 minutes and 30 seconds, and the faster model takes 13 minutes and 55 seconds[2]. It is possible to conclude that the dot product is a faster similarity metric than cosine when the embeddings are normalised.

6.1.3 Exact set match accuracy

The structuring rules are a crucial part of the proposed algorithm, as they are responsible to build the final query with every SQL clause identified and the respective values. The following evaluation component is to understand where the query formulation failed and can be improved. The model *multi-qa-MiniLM-L6-cos-v1* is selected using the dot product similarity metric to analyse the component match of reproduced queries to the target

[1] Sentence-transformers is a framework which allows users to compute dense vectors of words, paragraphs and images while providing open source state-of-the-art models fine-tuned on different tasks

[2] Every experiment was reproduced using a MacBook Pro laptop with an M1 pro CPU chip and 16 Gb of RAM

from the benchmark. The reproduced queries are subject to the evaluation script, where each is analysed for its SQL components.

The evaluation script developed by the benchmark authors lacks in accurately assessing the performance of the proposed algorithm in terms of the exact set matching as the queries are not correctly classified. For instance, the query "*SELECT farm.WorkingHorses FROM farm*" is not considered correct because the selected column includes the table name attached to it. To address this limitation, this work established a similar exact set match script, mapping every possible SQL clause that appears in the gold query. Table 6.2 represents the exact set match accuracy divided into three categories. The *simple* category comprises queries that do not involve joins or sub-selects, while the *join* category requires at least one join operation. The *sub-select* category involves only nested queries. Each category has query types divided by every combination that appears in the train and development sets.

Table 6.2: Customised, exact set match of the predicted queries for train and development sets with evey possible clause combination

Query	Query Type	Train		Dev	
		Total	**Correct**	**Total**	**Correct**
Simple	Select	81	70	22	21
	Where	55	33	19	13
	Where & Group By	2	2	0	0
	Where & Order By	2	0	0	0
	Order By	71	47	19	13
	Order By & Group By	31	23	8	7
	Group By	24	15	8	5
	Having & Group By	20	17	2	1
	Having & Group By & Where	1	1	0	0
Join	Simple Join	25	15	6	4
	Join & Order By	27	23	8	5
	Join & Group By	8	2	2	0
	Join & Order By & Group By	10	3	4	0
	Join & Group By & Having	16	4	7	5
	Join & Where	28	17	9	5
Sub-select	Simple Sub-Select	26	10	6	2
	Sub-Select & Where	2	0	0	0
		429	**282**	**120**	**81**

The number of correct queries classified by the execution accuracy totalised 298 and 85 for train and development, respectively. However, a comparison with the execution accuracy table indicates a decrease of 16 and 4 correct queries in the train and development sets, respectively. After analysing the mismatched queries, the following observations

were made:

- Ten queries from the train and two from development sets lacked the *ASC* clause. The questions related to these queries involved alphabetical ordering, and the proposed algorithm does not add this specific clause when the keywords related to *ASC* are not triggered. However, the absence of this clause did not impact the output as the engine that executes the query always orders the column alphabetically, which is attached to the *ORDER BY* clause.

- There are three queries from the train and one from development, which added an unnecessary *GROUP BY* clause. The presence of the keyword "*each*" in every question triggered an operation related to the grouping clause. For example, the question "Return the characters and durations for each actor." asks specifically to select the column by different actors. After analysing the question, the algorithm understands that it needs to add a grouping clause so the *actors* are differentiated. As the word *actors* refers to the table instead of the column, the algorithm attaches the table's primary key to the *GROUP BY* clause. Given that the output remained unaltered as there were no repeated values inside the primary key column.

- One of the questions from the train set refers to two different tables, leading the algorithm to execute a *JOIN* between them unnecessarily as they don't trigger the exception rules. However, the content remained unaltered since the selected columns belong to only one table, and the values between the primary and foreign keys are the same.

- The remaining cases in the train set consisted of one question where the missing *BETWEEN* clause was replaced with a double *WHERE* generating the same output. Another case was a missing *ORDER BY* where the algorithm failed to identify the column to order. However, the values of the column were initially ordered correctly, producing the same result. In the development set, the only mismatched question featured an additional *ASC* clause. It is odd as the question asked explicitly to order the column in ascending order, and the column is numerical. As in the previous case, the column was initially ordered, and both queries generated the correct output.

Among the issues discussed, only two have a significant impact. The first one is the unnecessary *JOIN* operation, which can lead to different results in different questions. The second issue involves the missing *ORDER BY* clause, which ensures output alignment with the user's specific order. The rest of the issues lead to the same output, even if some clauses are modified as adding the extra *ASC* or removing it and by that are not considered relevant.

Upon careful examination, it is possible to assess that the algorithm fails the most in replicating a specific type of join and nested queries. An analysis of the errors associated with each query type revealed the following findings:

- Simple Queries - In the category of simple, the most common failures were observed in the single *WHERE* and *ORDER BY* clauses. Difficulties arise in the "*WHERE*" clause when the algorithm fails to identify values to filter due to linguistic patterns that are not established in the algorithm. For instance, in the question

"*What are the names of actors who have been in the musical titled The Phantom of the Opera*", the algorithm could not identify that the pattern "*The Phantom of the Opera*" is related to a value because it is not enclosed inside the quotation marks and consists of a combination different than just proper nouns. Regarding the *ORDER BY* clause, there are ten cases where adding an *ASC* clause would render a correct query. Within the "*GROUP BY*" query type, some queries also failed to produce correct results due to a linguistic pattern where the term "*different*" is associated with both the "*DISTINCT*" clause and the "*GROUP BY*" clause. However, the vocabulary of the proposed algorithm categorises it as a "*DISTINCT*" clause, leading to inaccuracies.

- Join Queries - The biggest challenge of the *JOIN* queries focuses on the simple ones and the ones involving two additional clauses. When a question mentions multiple tables, the join module attempts to combine them together. However, the problem arises when the question has more tables mentioned than the ones necessary to concatenate, and they do not fall under the linguistic exceptions that can be disregarded. For example, the question "*Show the name and number of employees for the departments managed by heads whose temporary acting value is 'Yes'?*" identifies three tables "*departments*", "*heads*", and *"management"* while the table heads should be ignored. Additional issues emerge when the question requires a join and other clauses as *ORDER BY* and *GROUP BY*. The complexity of structuring an accurate query increases as more clauses are incorporated into it.

- Sub-Select - The *Sub-Select* clause is the least appearing clause in both sets, although it is also the most difficult to assemble. The linguistic patterns referred to this clause are challenging to separate from the others. For example, the word "*have*" can initiate three clauses: the *HAVING*, an *ORDER BY* or a *Sub-Select*. Given that difficulty, the algorithm struggles to identify correctly in which question it needs to add a *Sub-Select* clause.

Another issue that adds to these challenges is the overlap of structuring rules. The rules to assemble the query in the proposed algorithm are intrinsically related to the linguistic patterns. When the question turns out to be very long and diversified in terms of patterns that could be matched, some of them tend to overlap. For example, in the question "*Of all the contestants who got voted, what is the contestant number and name of the contestant who got least votes?*" where instead of selecting the column *contestantNumber*, the algorithm added a *COUNT* operator to the *SELECT* clause. This is a clear example where a keyword from the vocabulary triggered a specific operator and overlapped the selected column. Despite all the major difficulties, the proposed algorithm demonstrated the ability to correctly identify which columns to select and what operations to perform in a majority of the questions within both the train and the development sets.

6.2 Results on the entire Benchmark

In the previous section, the algorithm showed the capability to translate the plain language to SQL queries if the databases have correctly structured names of columns and tables. In this section, it is evaluated how the algorithm performs in the entire benchmark and which

are the biggest constraints to it. The entire dataset without *"EXCEPT"*, *"UNION"*, and *INTERSECT* clauses comprises 6470 and 958 questions from train and development sets.

6.2.1 Execution Accuracy and Time Execution

Table 6.3 displays the two best embedding models, which were chosen from the previous section, their execution accuracy in the train and development sets using two distinct similarity metrics, and the computational time. The main goal is to demonstrate the change in the computational time between two models and their similarity metrics and choose the best combination to follow.

Table 6.3: Execution accuracy comparing the two different models with two distinct similarity metrics and their computational time in the train and development sets

Pre-trained model	**Cosine Similarity**		**Dot Product Similarity**	
	Train	**Dev**	**Train**	**Dev**
multi-qa-mpnet-base-dot-v1	**26,19**	**28,63**	25,51	27,97
Computational Time	5h; 34m	43m; 9s	5h; 8m	43m; 51s
multi-qa-MiniLM-L6-cos-v1	25,08	27,37	24,79	27,27
Computational Time	4h; 5m	34m; 26s	**3h; 56m**	**34m; 20s**

After observing the results, it is possible to infer that the bigger model has produced more correct queries than the smallest one. However, the difference between the two models is not substantial. The major change occurred in the execution time of the selected models on the train set. As it is possible to observe, the biggest model takes 5 hours and 34 minutes to run the experiment using the cosine similarity. While the smallest model has a decrease of 1 hour and 29 minutes, totalising 4 hours and 5 minutes. The average calculation of time necessary to reproduce one question is 3.1 seconds for the most extensive model and 2.27 for the smallest one. In comparison to the dot product metric, the models took less time to run the experiment. As explained previously, the dot product metric is cheaper in computational time and is now clearly visible. However, the difference is not substantial, the significant change in time happened using the biggest model, where it decreased by 28 minutes in computational time. From the previous table 6.1, it was possible to extract that there was no difference in terms of execution accuracy between similarity metrics and the respective models. In this case, it is possible to visualise that the cosine similarity metric produced slightly better results than the dot product. The possible conclusion to this situation is that it is preferable to ignore the magnitude of the embeddings. The proposed algorithm only compares linguistic patterns with the names of the tables and columns to retrieve the one with the highest similarity. The linguistic patterns the model finds are mainly composed of one or two words in most cases, rarely going above these values. Given that cosine similarity is preferred and the use of the *multi-qa-MiniLM-L6-cos-v1* embedding model as this combination is the cost-effective approach and reproduces similar results compared to the best performer model.

6.3 Contrasts of the outcomes

The previous results are drastically different from the ones presented in section 6.1. After conducting an analysis of the errors, it was possible to conclude that there are two major constraints to achieve better results:

- Database Representation - The proposed algorithm consists of many rules which approach the databases without changing any column name or table. This means that, at the moment, one of the most important perspectives is that a database must have clear and simple names for columns and tables. Otherwise, it would be highly probable that the text-to-SQL algorithm would select erroneous columns or tables. These errors also could lead to wrong operations, for instance, the question "What is the count of distinct employees with certificates?" from the database "*flight1*". In this question, it is only necessary to count distinct's "*eid*" from the table certificate. However, the algorithm does not recognise the "*eid*" as an employee id column and proceeds to join the table *employee* and *certificates* on one common column.
- Natural Language Diversity - The *SPIDER* text-to-SQL benchmark has many diverse questions, paraphrased several times to augment the number of possible queries. Given that the proposed algorithm could be misled by the diversity of vocabulary present in the question and produce wrong queries.

Given the problems mentioned above, the text-to-SQL algorithm has better performance if the databases have correctly named columns and tables and if the requests made by users are not ambiguous or too narrow, as this will further confuse the algorithm. With both sections explained, this chapter concludes that given the performance and the results provided, two possible combinations exist to use the text-toSQL algorithm. If the performance is preferable over the cost-effectiveness of the algorithm, then it is safe to choose the *multi-qa-mpnet-base-dot-v1* as it can cover a broad range of domains in combination with cosine similarity. However, if the computational time is a relatively important factor, combining the *multi-qa-MiniLM-L6-cos-v1* with dot product similarity is the best option.

Chapter 7

Conclusions

Translating requests in plain language into SQL code is a challenging task that has not yet been finalised. Despite the progress of all the previous methods, there remain challenges for developing high-quality text-to-SQL algorithms.

The rule-based techniques created intelligent systems, but they were constrained to single domains and with a minimal scope of natural language questions that could be answered. With the advances in deep learning approaches, many algorithms were created to answer as many questions as possible. However, the computational time and the domain of the training data are constraints to these models. This study proposed combining different techniques to tackle the cross-domain adaptability and the computational resources to train the models. A combination of distinct embedding models and similarity metrics experiments were conducted to understand the impact on the proposed algorithm.

The experiment on the pre-selected databases from the benchmark in the section 6.1 allowed us to answer both of the research questions proposed in the introduction:

- RQ1 - The execution accuracy provided in the table 6.1 confirmed that pre-trained models play an important role in the proposed algorithm. The combination of the pre-trained models with the created linguistic rules showcased that the proposed text-to-SQL algorithm can sustainably find correct tables and columns. It was identified that the best model depends on the user's expectation from it, for instance, the model with the best ratio between accuracy and computational time is *multi-qa-MiniLM-L6-cos-v1*.

- RQ2 - The inspection that this work provided of the exact set match for query types in the table 6.2 confirmed that ruled-based techniques that were created had a successful combination with the handwritten vocabulary and could generate working queries. Given that it is possible to confirm that it is not necessary to train a model on huge quantities of data and use extensive computational resources to create a cross-domain model, which can generate correctly structured SQL queries.

Although the performance of the algorithm on the entire benchmark was not so successful, this work also explained the problems related to it. In conclusion, this work demonstrated a new approach that could lead to new ideas or innovations in this challenging task. The work was developed in co-operation with **Orbitas**[1], where the NLIDB was also successfully implemented.

[1] https://www.orbitas.ai/

7.1 Limitations of the Proposed Algorithm and Future Work

The main limitation of the text-to-SQL algorithm is the fact that it struggles to reproduce correct queries when dealing with databases that are not following the SQL naming conventions. Another issue is dealing with linguistic patterns that were not covered in vocabulary and the rule-based structure of the algorithm. Given these two major limitations, this work proposed several future work directions, such as:

- Create a General Entity-Value Recognition - One way to deal with linguistic patterns that were not covered in this work is to create an algorithm similar to the Named Entity Recognition but for databases. This type of algorithm would help in finding the correct column for the value presented in the question. For instance, if the user asks the question, "What is the year of creation of the AC/DC album?". The algorithm would immediately know that AC/DC is a value from the column *bands*. With that type of algorithm, it is possible to correct any error referred to the *WHERE* clause.

- Improve existing rules - The performance that this work provided confirms that the rules for structuring the queries do not overfit any type of query. However, there is still room for improvement as sometimes there is an overlap of rules, and the query is wrongly structured. It would be beneficial to create more generalistic rules or propose a new architecture for structuring the query instead of the rule-based one.

- Hybrid Approach - Another way to tackle unknown linguistic patterns is to employ a hybrid approach. A hybrid approach may consist of a training phase where the model would try to understand the user question and partition it into several clauses. For instance, the model may understand which specific part of the question is referring to the *SELECT* clauses and which to the filtering ones. The combination of this model with pre-trained models for semantic search and rule-based techniques for query structuring could generate a powerful model able to understand the necessary SQL clauses in the question and maintain its domain adaptability. The Rai et al. (2023) has proposed a component boundary marking where each question is broken into steps.

- Automatic Column Labelling - To address poor column naming, an interesting next step would be creating a suitable model that can analyse the names of the columns and propose changes if they are not clear or understandable.

- Other fundamental SQL operations - Create Read Update Delete (CRUD) are the four essential terms to operate a relational database. While this study only focused on the *Read* operation, a beneficial approach would extend the existing algorithm to the level of other actions.

- In context Question Answering - Currently, the proposed text-to-SQL algorithm can only answer direct requests and questions. However, requesting information on top of the outputs may be necessary for daily use. Given that, extending this capability would exploit new areas of research as it would tackle diverse challenges such as context memory.

This study has identified several areas for future work to enhance the performance and capabilities of the text-to-SQL algorithm and enhance its usability in practical applications.

7.2 Limitations of the SPIDER benchmark

The recent works which employ deep learning approaches for the tex-to-SQL task have very little or no human intervention in the translation method. This means that some of the errors from the academic benchmarks could be inputted directly into the models. Regarding the benchmark dataset that was used in the evaluation phase, there are several inconsistencies. The first one is wrong SQL queries, the table 7.1 demonstrates a few questions that were found during the analysis of the results. As it is possible to visualise, three of them fail in the *WHERE* and *GROUP BY* clauses, while one of them has a totally incomplete question or the SQL clause do not belong to the question. Although there are only four questions that have been discovered, as this dataset involved the human effort to assemble it, it is plausible to assume that there could be more wrong queries or incomplete questions.

Table 7.1: Inconsistencies of the SQL queries from the SPIDER benchmark

Question	Gold SQL	Comment
What are the advisors?	*SELECT advisor* *FROM Student* *GROUP BY advisor* *HAVING count(*) >= 2*	The question is incomplete or wrong SQL
What are the statuses and average populations of each city?	*SELECT Status , avg(Population)* *FROM city* *GROUP BY Status*	The question requires to group the column city and not status
Show the musical nominee with award ”Bob Fosse” or ”Cleavant Derricks”.	*SELECT Nominee* *FROM musical* *WHERE Award = ”Tony Award”* *OR Award = ”Cleavant Derricks”*	The WHERE clause must have a ”Bob Fosse” value instead of ”Tony Award”
Who are the nominees who were nominated for either of the Bob Fosse or Cleavant Derricks awards?	*SELECT Nominee* *FROM musical* *WHERE Award = ”Tony Award”* *OR Award = ”Cleavant Derricks”*	The WHERE clause must have a ”Bob Fosse” value instead of ”Tony Award”

Another issue that arises is the multiple possibilities to answer the questions. There are multiple questions from this benchmark that could be answered with other SQL queries. Given that evaluating the queries only on one target query is not totally correct. The zero-shot algorithms or the ones based on the rule-based approaches could suffer from this type of evaluation even if the produced queries have the same execution output.

References

Abbas, S., Khan, M. U., Lee, S. U.-J., Abbas, A., and Bashir, A. K. (2022). A review of nlidb with deep learning: findings, challenges and open issues. *IEEE Access*.

Arnold, A., Nallapati, R., and Cohen, W. W. (2007). A comparative study of methods for transductive transfer learning. In *Seventh IEEE international conference on data mining workshops (ICDMW 2007)*, pages 77–82. IEEE.

Axler, S. (1997). *Linear algebra done right*. Springer Science & Business Media.

Bergamaschi, S., Guerra, F., Interlandi, M., Trillo Lado, R., Velegrakis, Y., et al. (2013). Quest: a keyword search system for relational data based on semantic and machine learning techniques. *Proceedings of the VLDB Endowment*, 6:1222–1225.

Blunschi, L., Jossen, C., Kossman, D., Mori, M., and Stockinger, K. (2012). Soda: Generating sql for business users. *arXiv preprint arXiv:1207.0134*.

Bogin, B., Gardner, M., and Berant, J. (2019). Representing schema structure with graph neural networks for text-to-sql parsing. *arXiv preprint arXiv:1905.06241*.

Brunner, U. and Stockinger, K. (2021). Valuenet: A natural language-to-sql system that learns from database information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2177–2182. IEEE.

Cao, R., Chen, L., Chen, Z., Zhao, Y., Zhu, S., and Yu, K. (2021). Lgesql: line graph enhanced text-to-sql model with mixed local and non-local relations. *arXiv preprint arXiv:2106.01093*.

Chamberlin, D. D. and Boyce, R. F. (1974). Sequel: A structured english query language. In *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, pages 249–264.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. *arXiv preprint cmp-lg/9706022*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Elgohary, A., Hosseini, S., and Awadallah, A. H. (2020). Speak to your parser: Interactive text-to-sql with natural language feedback. *arXiv preprint arXiv:2005.02539*.

Feldman, S. (1999). Nlp meets the jabberwocky: Natural language processing in information retrieval. *ONLINE-WESTON THEN WILTON-*, 23:62–73.

Ganti, V., He, Y., and Xin, D. (2010). Keyword++ a framework to improve keyword search over entity databases. *Proceedings of the VLDB Endowment*, 3(1-2):711–722.

Grishman, R. (1986). *Computational linguistics: an introduction*. Cambridge University Press.

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.-G., Liu, T., and Zhang, D. (2019). Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

He, P., Mao, Y., Chakrabarti, K., and Chen, W. (2019). X-sql: reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*.

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., and Slocum, J. (1978). Developing a natural language interface to complex data. *ACM Transactions on Database Systems (TODS)*, 3(2):105–147.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huang, P.-S., Wang, C., Singh, R., Yih, W.-t., and He, X. (2018). Natural language to structured query generation via meta-learning. *arXiv preprint arXiv:1803.02400*.

Hui, B., Geng, R., Wang, L., Qin, B., Li, B., Sun, J., and Li, Y. (2022). S^2sql: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers. *arXiv preprint arXiv:2203.06958*.

Jamil, H. M. (2017). 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.

Johnson, T. (1984). Natural language computing: the commercial applications. *The Knowledge Engineering Review*, 1(3):11–23.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Kaufmann, E., Bernstein, A., and Fischer, L. (2007). Nlp-reduce: A naive but domainindependent natural language interface for querying ontologies. In *4th European Semantic Web Conference ESWC*, pages 1–2. Springer Berlin.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Li, F. and Jagadish, H. V. (2014). Nalir: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 709–712.

Li, H., Zhang, J., Li, C., and Chen, H. (2023a). Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*.

Li, J., Hui, B., Cheng, R., Qin, B., Ma, C., Huo, N., Huang, F., Du, W., Si, L., and Li, Y. (2023b). Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. *arXiv preprint arXiv:2301.07507*.

Liddy, E. D. (1998). Enhanced text retrieval using natural language processing. *Bulletin of the American Society for Information Science and Technology*, 24(4):14–16.

Liddy, E. D. (2001). Natural language processing.

Lin, K., Bogin, B., Neumann, M., Berant, J., and Gardner, M. (2019). Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.

Liu, A., Hu, X., Wen, L., and Yu, P. S. (2023). A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lyu, Q., Chakrabarti, K., Hathi, S., Kundu, S., Zhang, J., and Chen, Z. (2020). Hybrid ranking network for text-to-sql. *arXiv preprint arXiv:2008.04759*.

MacDonald, M. C., Pearlmutter, N. J., and Seidenberg, M. S. (1994). The lexical nature of syntactic ambiguity resolution. *Psychological review*, 101(4):676.

Majhadi, K. and Machkour, M. (2021). The history and recent advances of natural language interfaces for databases querying. In *E3S Web of Conferences*, volume 229, page 01039. EDP Sciences.

Manning, C. D. (2008). *Introduction to information retrieval*. Syngress Publishing,.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

McDonald, D. D. (1993). Issues in the choice of a source for natural language generation. *Computational Linguistics*, 19(1):191–197.

Melamud, O., Goldberger, J., and Dagan, I. (2016). context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Mitra, B., Craswell, N., et al. (2018). An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126.

Naseem, U., Razzak, I., Khan, S. K., and Prasad, M. (2021). A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–35.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Papamichail, A., Zarras, A. V., and Vassiliadis, P. (2020). Do people use naming conventions in sql programming? In *SOFSEM 2020: Theory and Practice of Computer Science: 46th International Conference on Current Trends in Theory and Practice of Informatics, SOFSEM 2020, Limassol, Cyprus, January 20–24, 2020, Proceedings 46*, pages 429–440. Springer.

Qi, J., Tang, J., He, Z., Wan, X., Zhou, C., Wang, X., Zhang, Q., and Lin, Z. (2022). Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *arXiv preprint arXiv:2205.06983*.

Qin, B., Hui, B., Wang, L., Yang, M., Li, J., Li, B., Geng, R., Cao, R., Sun, J., Si, L., et al. (2022). A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*.

Quamar, A., Efthymiou, V., Lei, C., Özcan, F., et al. (2022). Natural language interfaces to data. *Foundations and Trends® in Databases*, 11(4):319–414.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Rai, D., Wang, B., Zhou, Y., and Yao, Z. (2023). Improving generalization in language model-based text-to-sql semantic parsing: Two simple semantic boundary-based techniques. *arXiv preprint arXiv:2305.17378*.

Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Rodd, J., Gaskell, G., and Marslen-Wilson, W. (2002). Making sense of semantic ambiguity: Semantic competition in lexical access. *Journal of memory and language*, 46(2):245–266.

Saha, D., Floratou, A., Sankaranarayanan, K., Minhas, U. F., Mittal, A. R., and Özcan, F. (2016). Athena: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment*, 9(12):1209–1220.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Scholak, T., Schucher, N., and Bahdanau, D. (2021). Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv preprint arXiv:2109.05093*.

Shi, X., Liu, Q., Fan, W., Philip, S. Y., and Zhu, R. (2010). Transfer learning on heterogenous feature spaces via spectral transformation. In *2010 IEEE international conference on data mining*, pages 1049–1054. IEEE.

Simitsis, A., Koutrika, G., and Ioannidis, Y. (2008). Précis: from unstructured keywords as queries to structured databases as answers. *The VLDB Journal*, 17(1):117–149.

Slocum, J. (1984). Machine translation: its history, current status, and future prospects. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pages 546–561.

Song, D., Schilder, F., Smiley, C., Brew, C., Zielund, T., Bretz, H., Martin, R., Dale, C., Duprey, J., Miller, T., et al. (2015). Tr discover: A natural language interface for querying and analyzing interlinked datasets. In *International Semantic Web Conference*, pages 21–37. Springer.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. (2020). Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Tata, S. and Lohman, G. M. (2008). Sqak: doing more with keywords. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 889–902.

Togelius, J. and Yannakakis, G. N. (2023). Choose your weapon: Survival strategies for depressed ai academics. *arXiv preprint arXiv:2304.06035*.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394.

Usta, A., Karakayali, A., and Ulusoy, Ö. (2021). Dbtagger: Multi-task learning for keyword mapping in nlidbs using bi-directional recurrent neural networks. *arXiv preprint arXiv:2101.04226*.

Waltinger, U., Tecuci, D., Olteanu, M., Mocanu, V., and Sullivan, S. (2013). Usi answers: Natural language question answering over (semi-) structured industry data. In *Twenty-Fifth IAAI Conference*.

Wang, B., Shin, R., Liu, X., Polozov, O., and Richardson, M. (2019). Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. (2020). Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers.

Winograd, T. (1974). Five lectures on artificial intelligence. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE.

Wong, Y. W. and Mooney, R. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446.

Woods, W. A. (1973). 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.

Xu, X., Liu, C., and Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Xuan, K., Wang, Y., Wang, Y., Wen, Z., and Dong, Y. (2021). Sead: End-to-end text-to-sql generation with schema-aware denoising. *arXiv preprint arXiv:2105.07911*.

Yaghmazadeh, N., Wang, Y., Dillig, I., and Dillig, T. (2017). Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26.

Yu, T., Li, Z., Zhang, Z., Zhang, R., and Radev, D. (2018a). Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769*.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., et al. (2018b). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Zeng, J., Lin, X. V., Xiong, C., Socher, R., Lyu, M. R., King, I., and Hoi, S. C. (2020). Photon: a robust cross-domain text-to-sql system. *arXiv preprint arXiv:2007.15280*.

Zeng, L., Parthasarathi, S. H. K., and Hakkani-Tur, D. (2023). N-best hypotheses reranking for text-to-sql systems. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 663–670. IEEE.

Zhao, Y., Jiang, J., Hu, Y., Lan, W., Zhu, H., Chauhan, A., Li, A., Pan, L., Wang, J., Hang, C.-W., et al. (2022). Importance of synthesizing high-quality data for text-to-sql parsing. *arXiv preprint arXiv:2212.08785*.

Zhong, V., Xiong, C., and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Appendix A

Train and Development Questions

Table A.1: Table of questions in train and development sets

TRAIN
QUESTION
How many heads of the departments are older than 56 ?
List the name, born state and age of the heads of departments ordered by age.
List the creation year, name and budget of each department.
What are the maximum and minimum budget of the departments?
What is the average number of employees of the departments whose rank is between 10 and 15?
What are the names of the heads who are born outside the California state?
What are the distinct creation years of the departments managed by a secretary born in state 'Alabama'?
What are the names of the states where at least 3 heads were born?
In which year were most departments established?
Show the name and number of employees for the departments managed by heads whose temporary acting value is 'Yes'?
How many acting statuses are there?
How many departments are led by heads who are not mentioned?
What are the distinct ages of the heads who are acting?
Which department has more than 1 head at a time?
List the id, name and the number of heads.
Which head's name has the substring 'Ha'?
List the id and name.
How many farms are there?
Count the number of farms.
List the total number of horses on farms in ascending order.
What is the total horses record for each farm, sorted ascending?
What are the hosts of competitions whose theme is not "Aliens"?
Return the hosts of competitions for which the theme is not Aliens?
What are the themes of farm competitions sorted by year in ascending order?
Return the themes of farm competitions, sorted by year ascending.

Table A.1 continued from previous page

What is the average number of working horses of farms
with more than 5000 total number of horses?
Give the average number of working horses on farms
with more than 5000 total horses.
What are the maximum and minimum number of
cows across all farms.
Return the maximum and minimum number of cows across all farms.
How many different statuses do cities have?
Count the number of different statuses.
List official names of cities in descending order of population.
What are the official names of cities, ordered descending by population?
List the official name and status of the city with the largest population.
What is the official name and status of the city with the most residents?
Show the years and the official names of the host cities of competitions.
Give the years and official names of the cities of each competition.
Show the official names of the cities that have hosted more than one competition.
What are the official names of cities that have hosted more than one competition?
Show the status of the city that has hosted the greatest number of competitions.
What is the status of the city that has hosted the most competitions?
Please show the themes of competitions with host cities
having populations larger than 1000.
What are the themes of competitions that have corresponding
host cities with more than 1000 residents?
Please show the different statuses of cities and the average
population of cities with each status.
What are the statuses and average populations of each city?
Please show the different statuses, ordered by the number
of cities that have each.
Return the different statuses of cities, ascending by frequency.
List the most common type of Status across cities.
What is the most common status across all cities?
List the official names of cities that have not held any competition.
What are the official names of cities that have not
hosted a farm competition?
Find the official names of cities with population
bigger than 1500 or smaller than 500.
What are the official names of cities that have
population over 1500 or less than 500?
Show the census ranking of cities whose
status are not ”Village”.
What are the census rankings of cities that do
not have the status ”Village”?
How many books are there?
List the writers of the books in ascending alphabetical order.
List the titles of the books in ascending order of issues.

Table A.1 continued from previous page

What are the titles of the books whose writer is not ”Elaine Lee”?
What are the title and issues of the books?
What are the dates of publications in descending order of price?
What are the distinct publishers of publications
with price higher than 5000000?
List the publisher of the publication with the highest price.
List the publication dates of publications with 3 lowest prices.
Show the title and publication dates of books.
Show writers who have published a book with
price more than 4000000.
Show the titles of books in descending order of publication price.
Show publishers that have more than one publication.
Show different publishers together with the number
of publications they have.
Please show the most common publication date.
List the writers who have written more than one book.
List the titles of books that are not published.
What is the number of distinct publication dates?
How many distinct publication dates are there in our record?
Show the prices of publications whose publisher
is either ”Person” or ”Wiley”
How many actors are there?
Count the number of actors.
List the name of actors in ascending alphabetical order.
What are the names of actors, ordered alphabetically?
What are the characters and duration of actors?
Return the characters and durations for each actor.
List the name of actors whose age is not 20.
What are the names of actors who are not 20 years old?
What are the characters of actors in descending order of age?
Return the characters for actors, ordered by age descending.
What is the duration of the oldest actor?
Return the duration of the actor with the greatest age.
What are the names of musicals with nominee ”Bob Fosse”?
Return the names of musicals who have the nominee Bob Fosse.
What are the distinct nominees of the musicals with
the award that is not ”Tony Award”?
Return the different nominees of musicals that have
an award that is not the Tony Award.
Show names of actors and names of musicals they are in.
What are the names of actors and the musicals that they are in?
Show names of actors that have appeared in musical
with name ”The Phantom of the Opera”.
What are the names of actors who have been in the
musical titled The Phantom of the Opera?

Table A.1 continued from previous page

Show names of actors in descending order of the
year their musical is awarded.
What are the names of actors ordered descending by the
year in which their musical was awarded?
Show names of musicals and the number of actors
who have appeared in the musicals.
How many actors have appeared in each musical?
Show names of musicals which have at least three actors.
What are the names of musicals who have at 3 or more actors?
Show different nominees and the number of musicals
they have been nominated.
How many musicals has each nominee been nominated for?
Please show the nominee who has been nominated
the greatest number of times.
Who is the nominee who has been nominated for the most musicals?
List the most common result of the musicals.
Return the most frequent result across all musicals.
List the nominees that have been nominated more than two musicals.
Who are the nominees who have been nominated more than two times?
List the name of musicals that do not have actors.
What are the names of musicals who have no actors?
Show the musical nominee with award ”Bob Fosse”
or ”Cleavant Derricks”.
Who are the nominees who were nominated for either
of the Bob Fosse or Cleavant Derricks awards?
How many editors are there?
List the names of editors in ascending order of age.
What are the names and ages of editors?
List the names of editors who are older than 25.
Show the names of editors of age either 24 or 25.
What is the name of the youngest editor?
What are the different ages of editors? Show each age
along with the number of editors of that age.
Please show the most common age of editors.
Show the distinct themes of journals.
Show the names of editors and the theme of journals
for which they serve on committees.
For each journal_committee, find the editor name
and the journal theme.
Show the names and ages of editors and the theme of
journals for which they serve on committees,
in ascending alphabetical order of theme.
Show the names of editors that are on the committee
of journals with sales bigger than 3000.

Table A.1 continued from previous page

Show the id, name of each editor and the number
of journal committees they are on.
Show the names of editors that are on at least two journal committees.
List the names of editors that are not on any journal committee.
What is the average sales of the journals that have an
editor whose work type is 'Photo'?
How many tracks do we have?
Count the number of tracks.
Show the name and location for all tracks.
What are the names and locations of all tracks?
Show names and seatings, ordered by seating for
all tracks opened after 2000.
What are the names and seatings for all tracks
opened after 2000, ordered by seating?
What is the name, location and seating for the
most recently opened track?
Return the name, location, and seating of the track
that was opened in the most recent year.
What is the minimum, maximum, and
average seating for all tracks.
Return the minimum, maximum, and average
seating across all tracks.
Show the name, location, open year for all tracks
with a seating higher than the average.
What are the names, locations, and years of opening
for tracks with seating higher than average?
What are distinct locations where tracks are located?
Give the different locations of tracks.
How many races are there?
Count the number of races.
What are the distinct classes that races can have?
Return the different classes of races.
Show name, class, and date for all races.
What are the names, classes, and dates for all races?
Show the race class and number of races in each class.
What are the different classes of races, and
how many races correspond to each?
What is the race class with most number of races.
Give the class of races that is most common.
List the race class with at least two races.
What are the classes of races that have two
or more corresponding races?
Show all track names that have had no races.
Return the names of tracks that have no had any races.

Table A.1 continued from previous page

Show year where a track with a seating at least
5000 opened and a track with seating
no more than 4000 opened.
What are the years of opening for tracks
with seating between 4000 and 5000?
Show the name of track and the number of
races in each track.
What are the names of different tracks,
and how many races has each had?
Show the name of track with most number of races.
What is the name of the track that has had
the greatest number of races?
Show the name and date for each race and its track name.
What are the names and dates of races, and the
names of the tracks where they are held?
Show the name and location of track with 1 race.
What are the names and locations of tracks
that have had exactly 1 race?
How many climbers are there?
Count the number of climbers.
List the names of climbers in descending order of points.
What are the names of the climbers, ordered by points descending?
List the names of climbers whose country is not Switzerland.
What are the names of climbers who are not
from the country of Switzerland?
What is the maximum point for climbers whose
country is United Kingdom?
Return the maximum number of points for
climbers from the United Kingdom.
How many distinct countries are the climbers from?
Count the number of different countries that climbers are from.
What are the names of mountains in ascending alphabetical order?
Give the names of mountains in alphabetical order.
What are the countries of mountains with height bigger than 5000?
Return the countries of the mountains that have a height larger than 5000.
What is the name of the highest mountain?
Return the name of the mountain with the greatest height.
List the distinct ranges of the mountains with the top 3 prominence.
What are the different ranges of the 3 mountains with the highest prominence?
Show names of climbers and the names of mountains they climb.
What are the names of climbers and the corresponding
names of mountains that they climb?
Show the names of climbers and the heights of mountains they climb.
What are the names of climbers and the corresponding
heights of the mountains that they climb?

Table A.1 continued from previous page

Show the height of the mountain climbed by
the climber with the maximum points.
What is the height of the mountain climbined
by the climbing who had the most points?
Show the distinct names of mountains climbed
by climbers from country ”West Germany”.
What are the different names of mountains ascended
by climbers from the country of West Germany?
Show the times used by climbers to climb mountains
in Country Uganda.
What are the times used by climbers who climbed
mountains in the country of Uganda?
Please show the countries and the number of
climbers from each country.
How many climbers are from each country?
List the countries that have more than one mountain.
Which countries have more than one mountain?
List the names of mountains that do not have any climber.
What are the names of countains that no climber has climbed?
Show the range that has the most number of mountains.
Which range contains the most mountains?
Show the names of mountains with height
more than 5000 or prominence more than 1000.
What are the names of mountains that have a h
eight of over 5000 or a prominence of over 1000?
How many body builders are there?
List the total scores of body builders in ascending order.
List the snatch score and clean jerk score of
body builders in ascending order of snatch score.
What is the average snatch score of body builders?
What are the clean and jerk score of the body builder
with the highest total score?
What are the birthdays of people in ascending order of height?
What are the names of body builders?
What are the names of body builders whose
total score is higher than 300?
What is the name of the body builder
with the greatest body weight?
What are the birth date and birth place of the
body builder with the highest total points?
What are the heights of body builders with
total score smaller than 315?
What is the average total score of body builders
with height bigger than 200?

Table A.1 continued from previous page

What are the names of body builders in descending
order of total scores?
List each birth place along with the number of people from there.
What is the most common birth place of people?
What are the birth places that are shared by at least two people?
List the height and weight of people in descending order of height.
Show all information about each body builder.
How many distinct birth places are there?
How many persons are not body builders?
List the weight of the body builders who have snatch
score higher than 140 or have the height greater than 200.
What are the total scores of the body builders
whose birthday contains the string ”January” ?
What is the minimum snatch score?
How many elections are there?
List the votes of elections in descending order.
List the dates and vote percents of elections.
What are the minimum and maximum vote percents of elections?
What are the names and parties of representatives?
What are the names of representatives whose
party is not ”Republican”?
What are the life spans of representatives from
New York state or Indiana state?
What are the names of representatives and the
dates of elections they participated in.
What are the names of representatives with more
than 10000 votes in election?
What are the names of representatives in descending
order of votes?
What is the party of the representative that has
the smallest number of votes.
What are the lifespans of representatives in
descending order of vote percent?
What is the average number of votes of representatives
from party ”Republican”?
What are the different parties of representative?
Show the party name and the number of representatives in each party.
What is the party that has the largest number of representatives?
What parties have at least three representatives?
What states have at least two representatives?
List the names of representatives that have not
participated in elections listed here.
How many distinct parties are there for representatives?
How many performances are there?
List the hosts of performances in ascending order of attendance.

Table A.1 continued from previous page

What are the dates and locations of performances?
Show the attendances of the performances at
location ”TD Garden” or ”Bell Centre”
What is the average number of attendees for performances?
What is the date of the performance with
the highest number of attendees?
Show different locations and the number
of performances at each location.
Show the most common location of performances.
Show the locations that have at least two performances.
Show the names of members and the location
of the performances they attended.
Show the names of members and the location of
performances they attended in ascending
alphabetical order of their names.
Show the dates of performances with
attending members whose roles are ”Violin”.
Show the names of members and the dates of
performances they attended in descending order
of attendance of the performances.
List the names of members who did not attend any performance.
How many gymnasts are there?
Count the number of gymnasts.
List the total points of gymnasts in descending order.
What are the total points for all gymnasts,
ordered by total points descending?
List the total points of gymnasts in descending
order of floor exercise points.
What are the total points of gymnasts,
ordered by their floor exercise points descending?
What is the average horizontal bar points for all gymnasts?
Return the average horizontal bar points across all gymnasts.
What are the names of people in ascending alphabetical order?
Return the names of people, ordered alphabetically.
What are the names of gymnasts?
Return the names of the gymnasts.
What are the names of gymnasts whose
hometown is not ”Santo Domingo”?
Return the names of gymnasts
who did not grow up in Santo Domingo.
What is the age of the tallest person?
Return the age of the person with the greatest height.
List the names of the top 5 oldest people.
What are the names of the five oldest people?
What is the total point count of the youngest gymnast?

Table A.1 continued from previous page

Return the total points of the gymnast with the lowest age.
What is the average age of all gymnasts?
Return the average age across all gymnasts.
What are the distinct hometowns of gymnasts
with total points more than 57.5?
Give the different hometowns of gymnasts
that have a total point score of above 57.5.
What are the hometowns of gymnasts and
the corresponding number of gymnasts?
How many gymnasts are from each hometown?
What is the most common hometown of gymnasts?
Return the hometown that is most common among gymnasts.
What are the hometowns that are shared by at least two gymnasts?
Give the hometowns from which two or more gymnasts are from.
List the names of gymnasts in ascending order by their heights.
What are the names of gymnasts, ordered by their heights ascending?
How many distinct hometowns did these people have?
Count the number of different hometowns of these people.
Show the ages of gymnasts in descending order of total points.
What are the ages of the gymnasts, ordered
descending by their total points?
How many wrestlers are there?
Count the number of wrestlers.
List the names of wrestlers in descending order of days held.
What are the names of the wrestlers, ordered descending by days held?
What is the name of the wrestler with the fewest days held?
Return the name of the wrestler who had the
lowest number of days held.
What are the distinct reigns of wrestlers
whose location is not ”Tokyo,Japan” ?
Give the different reigns of wrestlers who
are not located in Tokyo, Japan.
What are the names and location of the wrestlers?
Give the names and locations of all wrestlers.
What are the elimination moves of wrestlers
whose team is ”Team Orton”?
Return the elimination movies of
wrestlers on Team Orton.
What are the names of wrestlers
and the elimination moves?
Give the names of wrestlers and
their elimination moves.
List the names of wrestlers and the teams
in elimination in descending order of days held.

Table A.1 continued from previous page

What are the names of wrestlers and their
teams in elimination, ordered descending by days held?
List the time of elimination of the wrestlers
with largest days held.
What is the time of elimination for the wrestler
with the most days held?
Show times of elimination of wrestlers
with days held more than 50.
What are the times of elimination for
wrestlers with over 50 days held?
Show different teams in eliminations and the
number of eliminations from each team.
How many eliminations did each team have?
Show teams that have suffered more than three eliminations.
Which teams had more than 3 eliminations?
Show the reign and days held of wrestlers.
What are the reigns and days held of all wrestlers?
What are the names of wrestlers days held less than 100?
Return the names of wrestlers with fewer than 100 days held.
Please show the most common reigns of wrestlers.
Which reign is the most common among wrestlers?
List the locations that are shared by more than two wrestlers.
Which locations are shared by more than two wrestlers?
List the names of wrestlers that have not been eliminated.
What are the names of wrestlers who have never been eliminated?
What is the number of distinct teams that suffer elimination?
How many different teams have had eliminated wrestlers?
Show the times of elimination by ”Punk” or ”Orton”.
What are the times of elimination for
any instances in which the elimination
was done by Punk or Orton?
How many devices are there?
Count the number of devices.
List the carriers of devices in ascending alphabetical order.
What are the different carriers for devices,
listed in alphabetical order?
What are the carriers of devices whose
software platforms are not ”Android”?
Return the device carriers that do not have
Android as their software platform.
What are the names of shops in ascending
order of open year?
Return the names of shops, ordered by
year of opening ascending.
What is the average quantity of stocks?

Table A.1 continued from previous page

Give the average quantity of stocks.
What are the names and location of the shops
in ascending alphabetical order of name.
Return the names and locations of shops,
ordered by name in alphabetical order.
How many different software platforms are there for devices?
Count the number of different software platforms.
List the open date of open year of the shop named ”Apple”.
What are the open dates and years for the shop named Apple?
List the name of the shop with the latest open year.
What is the shop name corresponding to the
shop that opened in the most recent year?
Show names of shops and the carriers of devices they have in stock.
What are the names of device shops, and what
are the carriers that they carry devices in stock for?
Show names of shops that have more than one
kind of device in stock.
What are the names of shops that have more than a
single kind of device in stock?
Show the name of the shop that has the most
kind of devices in stock.
What is the name of the shop that has the most
different kinds of devices in stock?
Show the name of the shop that have the largest quantity
of devices in stock.
What is the name of the shop that has the greatest quantity
of devices in stock?
Please show different software platforms and the corresponding
number of devices using each.
What are the different software platforms for devices, and how
many devices have each?
Please show the software platforms of devices in
descending order of the count.
What are the different software platforms for devices,
ordered by frequency descending?
List the software platform shared by the greatest
number of devices.
What is the software platform that is most common
amongst all devices?
List the names of shops that have no devices in stock.
What are the names of shops that do not have any devices in stock?
List the carriers of devices that have no devices in stock.
What are the carriers of devices that are not in stock anywhere?
Show the carriers of devices in stock at more than one shop.

Table A.1 continued from previous page

What are the carriers of devices that are in stock
in more than a single shop?
How many different captain ranks are there?
Count the number of different ranks of captain.
How many captains are in each rank?
Count the number of captains that have each rank.
How many captains with younger than 50 are in each rank?
Count the number of captains younger than 50 of each rank.
Sort all captain names by their ages from old to young.
What are the names of captains, sorted by age descending?
Find the name, class and rank of all captains.
What are the names, classes, and ranks of all captains?
Which rank is the most common among captains?
Return the rank for which there are the fewest captains.
Which classes have more than two captains?
Give the classes that have more than two captains.
Find the name of captains whose rank are
either Midshipman or Lieutenant.
What are the names of captains that have either
the rank Midshipman or Lieutenant?
What are the average and minimum age
of captains in different class?
Return the average and minimum age
of captains in each class.
What is the name of the youngest captain?
Return the name of the youngest captain.
how many ships are there?
Count the number of ships.
Find the name, type, and flag of the ship that
is built in the most recent year.
What is the name, type, and flag of the ship
that was built in the most recent year?
Group by ships by flag, and return number
of ships that have each flag.
What are the different ship flags, and how
many ships have each?
Which flag is most widely used among all ships?
Return the flag that is most common among all ships.
List all ship names in the order of built year and class.
What are the names of ships, ordered by year
they were built and their class?
In which year were most of ships built?
What is the year in which most ships were built?
Find the name of the ships that have more than one captain.
What are the names of ships that have more than one captain?

Table A.1 continued from previous page

what are the names and classes of the ships that do not have any captain yet?
Return the names and classes of ships that do not have a captain?
Find the name of the ship that is steered by the youngest captain.
What is the name of the ship that is commanded by the youngest captain?
Find the name and flag of ships that are not steered by any captain with Midshipman rank.
What are the names and flags of ships that do not have a captain with the rank of Midshipman?
How many drivers are there?
Show the name, home city, and age for all drivers.
Show the party and the number of drivers in each party.
Show the name of drivers in descending order of age.
Show all different home cities.
Show the home city with the most number of drivers.
Show the party with drivers from Hartford and drivers older than 40.
Show home city where at least two drivers older than 40 are from.
Show the names of the drivers without a school bus.
Show the types of schools that have two schools.
Show the school name and driver name for all school buses.
What is the maximum, minimum and average years spent working on a school bus?
Show the school name and type for schools without a school bus.
Show the type of school and the number of buses for each type.
How many drivers are from Hartford city or younger than 40?
List names for drivers from Hartford city and younger than 40.
find the name of driver who is driving the school bus with the longest working history.
DEVELOPMENT
Question
How many poker players are there?
Count the number of poker players.
List the earnings of poker players in descending order.
What are the earnings of poker players, ordered descending by value?
List the final tables made and the best finishes of poker players.
What are the final tables made and best finishes for all poker players?
What is the average earnings of poker players?
Return the average earnings across all poker players.
What is the money rank of the poker player with the highest earnings?
Return the money rank of the player with the greatest earnings.
What is the maximum number of final tables made among poker players with earnings less than 200000?

Table A.1 continued from previous page

Return the maximum final tables made across all
poker players who have earnings below 200000.
What are the names of poker players?
Return the names of all the poker players.
What are the names of poker players whose
earnings is higher than 300000?
Give the names of poker players who have
earnings above 300000.
List the names of poker players ordered by the final
tables made in ascending order.
What are the names of poker players, ordered ascending
by the number of final tables they have made?
What is the birth date of the poker player with the lowest earnings?
Return the birth date of the poker player with the lowest earnings.
What is the money rank of the tallest poker player?
Return the money rank of the poker player with the greatest height.
What is the average earnings of poker players with height higher than 200?
Give average earnings of poker players who are taller than 200.
What are the names of poker players in descending order of earnings?
Return the names of poker players sorted by their earnings descending.
What are different nationalities of people and the
corresponding number of people from each nation?
How many people are there of each nationality?
What is the most common nationality of people?
Give the nationality that is most common across all people.
What are the nationalities that are shared by at least two people?
Return the nationalities for which there are two or more people.
List the names and birth dates of people in
ascending alphabetical order of name.
What are the names and birth dates of people,
ordered by their names in alphabetical order?
Show names of people whose nationality is not ”Russia”.
What are the names of people who are not from Russia?
List the names of people that are not poker players.
What are the names of people who do not play poker?
How many distinct nationalities are there?
Count the number of different nationalities.
How many states are there?
List the contestant numbers and names,
ordered by contestant name descending.
List the vote ids, phone numbers and states of all votes.
What are the maximum and minimum values of area codes?
What is last date created of votes from the state ’CA’?
What are the names of the contestants whose
names are not ’Jessie Alloway’

Table A.1 continued from previous page

What are the distinct states and create time of all votes?
What are the contestant numbers and names of the
contestants who had at least two votes?
Of all the contestants who got voted, what is the
contestant number and name of the contestant who
got least votes?
What are the number of votes from state
'NY' or 'CA'?
How many contestants did not get voted?
What is the area code in which the most voters voted?
What are the create dates, states, and phone numbers
of the votes that were for the contestant
named 'Tabatha Gehling'?
Return the names of the contestants whose
names contain the substring 'Al' .
How many conductors are there?
Count the number of conductors.
List the names of conductors in ascending order of age.
What are the names of conductors, ordered by age?
What are the names of conductors
whose nationalities are not "USA"?
Return the names of conductors that
do not have the nationality "USA".
What are the record companies of orchestras in
descending order of years in which they were founded?
Return the record companies of orchestras, sorted
descending by the years in which they were founded.
What is the average attendance of shows?
Return the average attendance across all shows.
What are the maximum and minimum share
of performances whose type is not "Live final".
Return the maximum and minimum shares
for performances that do not have the type "Live final".
How many different nationalities do conductors have?
Count the number of different nationalities of conductors.
List names of conductors in descending order of years of work.
What are the names of conductors, sorted
descending by the number of years they have worked?
List the name of the conductor with the most years of work.
What is the name of the conductor who
has worked the greatest number of years?
Show the names of conductors and the
orchestras they have conducted.
What are the names of conductors as well
as the corresonding orchestras that they have conducted?

Table A.1 continued from previous page

Show the names of conductors that have
conducted more than one orchestras.
What are the names of conductors who have
conducted at more than one orchestra?
Show the name of the conductor that has
conducted the most number of orchestras.
What is the name of the conductor who has
conducted the most orchestras?
Please show the name of the conductor that
has conducted orchestras founded after 2008.
What are the names of conductors who have
conducted orchestras founded after the year 2008?
Please show the different record companies
and the corresponding number of orchestras.
How many orchestras does each record company manage?
Please show the record formats of orchestras
in ascending order of count.
What are the major record formats of
orchestras, sorted by their frequency?
List the record company shared by
the most number of orchestras.
What is the record company used by
the greatest number of orchestras?
List the names of orchestras that have no performance.
What are the orchestras that do not have any performances?
Find the number of orchestras whose
record format is ”CD” or ”DVD”.
Count the number of orchestras that have
CD or DVD as their record format.
Show the years in which orchestras that
have given more than one performance are founded.
What are years of founding for orchestras
that have had more than a single performance?
How many singers are there?
What is the count of singers?
List the name of singers in ascending order of net worth.
What are the names of singers ordered by ascending net worth?
What are the birth year and citizenship of singers?
What are the birth years and citizenships of the singers?
List the name of singers whose citizenship is not ”France”.
What are the names of the singers who are not French citizens?
Show the name of singers whose birth
year is either 1948 or 1949?
What are the names of the singers whose
birth years are either 1948 or 1949?

Table A.1 continued from previous page

What is the name of the singer with the largest net worth?
What is the name of the singer who is worth the most?
Show different citizenship of singers and the number of singers of each citizenship.
For each citizenship, how many singers are from that country?
Please show the most common citizenship of singers.
What is the most common singer citizenship ?
Show different citizenships and the maximum net worth of singers of each citizenship.
For each citizenship, what is the maximum net worth?
Show titles of songs and names of singers.
What are the song titles and singer names?
Show distinct names of singers that have songs with sales more than 300000.
what are the different names of the singers that have sales more than 300000?
Show the names of singers that have more than one song.
What are the names of the singers that have more than one songs?
Show the names of singers and the total sales of their songs.
For each singer name, what is the total sales for their songs?
List the name of singers that do not have any song.
What is the sname of every sing that does not have any song?

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa