NOVA IMS Information Management School

MDSAA

Master Degree Program in

Data Science and Advanced Analytics

Context-driven Semantic Parsing to expand cross-domain Text-to-SQL

Inês Daniela Cardoso Nascimento

Master Thesis

presented as partial requirement for obtaining a 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

Context-driven Semantic Parsing to expand cross-domain Text-to-SQL

by

Inês Daniela Cardoso Nascimento

Master Thesis presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Data Science.

Supervised by

Mauro Castelli, PhD, NOVA IMS

Co-Supervised by

Fernando Peres

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.

Inês Nascimento

Lisbon, 20 of November of 2024

Acknowledgements

I want to thank Mauro Castelli and Fernando Peres, supervisor, and co-supervisor for guiding me throughout all the thesis construction steps. Also, I want to thank Mauro Castelli, Fernando Peres, and Yuriy for encouraging me to develop this thesis on such an innovative topic as natural learning processing making this thesis experience very interesting and enthusiastic to develop. Moreover, I would like to thank Yuriy especially for learning a lot from him and for being very present.

I would like to express my eternal gratitude to the most beautiful human beings I know, my parents, for being unconditional supporters of my whole life, your motivation and the fact that both of you believed in my abilities made me a braver person.

Thank you to my sister and brother-in-law for helping me to rest during this period of work and study, making this stage of my life slightly easier without losing focus.

Thank you to all the people I work with for having enormous patience with me, even on days when I lost many hours of sleep and had to work. Thank you for all the trust, words of support, and the healthy work environment.

Thank you to my master's colleagues who were also supervised during the thesis in the same group as me, allowing for a good environment and mutual help.

To my friends, thank you for believing in me and my work and for making an effort to fill me with positivity so that I could successfully complete this stage of my life.

Finally, an enormous thank you to all the professors at NOVA IMS who contributed to consolidating all the concepts of data science and advanced analytics, which are certainly implicitly demonstrated throughout this thesis.

Abstract

This research study focuses on applying sequence-to-sequence models to approach conversational text-to-SQL by comparing different methodologies. This study proposes a pre-training in the T5-base model with *WikiSQL* data later fine-tuned with *SParC* data, which involves taxonomy, also known as schema linking and tree dependency parsing integrated. This model was later compared through an ablation study with training *SParC* data with a pre-train in T5-base model with fine-tuning, where all procedure was kept except the differentiations on the model development itself. The impact of taxonomy and dependency parsing were checked through model results. These methodologies were tested through four samples defined in advance using different database domains in a way that all benchmark was trained and tested. The metrics used were the execution with values and the exact set match without values that evaluates the capacity of the queries to access the database and bring a value or build the query structure. Thus, computational runtime and proper machines were described in order to evaluate the impact of the final result. The computational power challenges found suggests that future work requires to be developed using this alternative approach.

Keywords

Natural Language Processing, Conversational text-to-SQL, Schema linking, Dependency parsing

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

BOW Bag of words

DB Database

Dev Development

EM Exact Match

EX Execution Accuracy

FN False Negatives

FP False Positives

GNN Graph Neural Network

IEM Interaction-level Exact Match

IEX Interaction-level Execution accuracy

LLM Large Language Models

LSTM Long Short-Term Memory

ML Machine Learning

NER Named Entity Recognition

NL Natural Language

NLG Natural Language Generation

NLP Natural Language Processing

NLQ Natural Language Query

NLU Natural Language Understanding

PCMF1 Partial Component Match F1

POS Part-of-speech tagging

RASAT Relation-Aware Self-Attention-augmented T5

RAT Relation-Aware Transformer

RDBMS Relational Database Management Systems

RNN Recurrent Neural Networks

Seq2seq Sequence-to-sequence

Seq2SQL Sequence-to-SQL

SFT Supervised Fine-Tuning

SQL Structured Query Language

SST Schema State Tracking

TF-IDF Term Frequency - Inverse Document Frequency

TN True Negatives

TP True Positives

UDT Utterance Dependency Tracking

Chapter 1

Introduction

Natural Language Processing (NLP) is a vast area that includes text-to-SQL, translating human input into structured query language (SQL) instructions. This enables even the people who do not know how to code in SQL to have access to the information in the databases (DBs) by asking a question allowing faster responses. This approach provides free expertise in SQL programming, thereby making communication and connection with databases more straightforward (Sui et al. (2023)). Via NLP, the model is able to analyze and contextualize user utterances that are presented in human natural language (NL) while SQL is the common programming language to deal with relational databases. In order to produce precise, syntactical, and logical SQL queries, the procedure entails parsing and gathering important information from the questions. Conversational text-to-SQL performance is measured by the exact match (EM), where the SQL query produced at the end of the model correctly matches a predetermined accurate query, and interaction match, where it assesses the capacity to keep context during a sequence of connected questions during a dialog. Until today's development, there have been many limitations regarding ambiguity and correct contextualized understanding of users' intentions. In recent years, pre-trained large language models (LLM) have brought more efficient ways to approach conversational text-to-SQL. Consequently, significant progress has been made in developing major user-friendly solutions for natural language as the original input for database (DB) queries (Sathick and Jaya (2015)). This change represents a turning point in the development of conversational text-to-SQL techniques, which are explored in this investigation. Moreover, to address all the concerns belonging to the current development of conversational text-to-SQL, it is essential to highlight challenges with major impact and focus on finding solutions. In the follow-up, the current manifested limitations are explained below:

• **Input Encoding** – The input suffers from the limits of serialization methods with token restrictions in terms of length. Due to the protection of the dialog context using a pre-trained LLM, serialized inputs struggle with encoding large database schemas.

• **Output Decoding** – Sequence-based techniques have difficulty avoiding syntactical errors, e.g. Seq2SQL, which might result in SQL code that has the wrong column and table names. Also, neural networks operating as "black-box" in sketch-based models make comprehending the models' decision-making procedures difficult, e.g. CatSQL (Fu

et al. (2023)).

• **Ambiguity** – Grammatical and paraphrasing problems while verifying the difficulties of managing unclear users' natural language questions (Wang et al. (2023)). Beyond these issues, the text-to-SQL encounters challenges with cross-domain adaptation to all existing benchmarks and context-dependent ambiguity, which reinforces steady progress due to the specific constraints of each benchmark and the heterogeneous nature of the data (Katsogiannis-Meimarakis (2023)).

• **Schema linking** – Drawbacks regarding a variety of different methods, including the identification of key candidate words and their main match within the taxonomy (Katsogiannis-Meimarakis and Koutrika (2023)). Finally, the requirement needs to be improved for accurate classification into value links, tables, or columns.

1.1 Main objectives

In this research study, the main objective of this master's thesis is to show a different approach to conversational text-to-SQL, therefore it is expected that the existing limitations will be reduced with the improvement of schema linking, where columns and tables are connected with a robust map of words to utterances being crucial to semantic parsing. As a consequence, ambiguity will be reduced and the context relationship in the dialogue will be improved. It is expected to introduce a hybrid model sensitive to both types of language, NLP text and SQL code, using the respective weights of the pre-trained models considering the context and various NLP techniques to accomplish top-tier performance. In the end, the answer to the research question will be given: How can a pre-train model be used to help the development of semantic parsing and taxonomy improvements in a large-scale cross-domain context-dependent dataset? This enables the improvement of current approaches by transposing them to real DB without losing their efficiency. It is also important to note that major computational challenges are part of text-to-SQL as well as other limitations shown by Kumar et al. (2022), Katsogiannis-Meimarakis and Koutrika (2023), Qin et al. (2022), and more.

In recent years, the exponential and incredible progress of large-scale pre-trained models has become part of many people's daily lives with the appearance of pre-trained models such as GPT-4 (Liu et al. (2023)). Therefore, the use of pre-trained models becomes essential making this study competitive with other studies.

1.2 Structure

The document is divided into chapters, it starts with an introduction of the main objectives of this master thesis focusing on the contributions in the field of text-to-SQL research

area. The other chapters present the following content:

• **Chapter 2** explains with some detail the important concepts that support conversational text-to-SQL.

• **Chapter 3** represents major higher-score projects developed in the area and their respective benchmark and drawbacks found until now.

• **Chapter 4** clarifies methodology implemented and tested as the solution to conversational text-to-SQL concerns.

• **Chapter 5** explains the built algorithm and its key characteristics.

• **Chapter 6** shows the final results of the implemented algorithm provided by leaderboard metrics adopted to compare to other works already developed.

• **Chapter 7** presents the conclusions and limitations found during all research and possible future work.

Chapter 2

Background

2.1 An Evolution of Natural Language Processing

Over the years, the term Natural Language Processing (NLP) appeared as a branch of Artificial Intelligence (AI) and Linguistics expressing and generating meaningful Human language from complex computational reasoning (Khurana et al. (2023)). Intuitively, NLP urge as an interdisciplinary field around the 1950s without being called by those terms, instead, the most popular languages were English and Russian at that time when machine translation started to be researched (Nadkarni et al. (2011)).

After all the speculation and money invested, the realistic limitations of machine translation emerged about a decade later. As such, it was created a research board called ALPAC, whereas extending processing time was one of the suggestions made. In general, ALPAC's problem was that they did not realize that the first experiments were only elementary and those only focused on a small sample rather than the complex reality (Hutchins (1994); Hutchins (2004)).

Furthermore, in the 1970s, Prolog, a logic programming language was created to support NLP applications (Colmerauer and Roussel (1996);Chowdhary (2020)), improving grammar beginning with a sentence's overall structure and gradually fragmenting it to understand it in smaller chunks, also known as top-down parsing. Moreover, in the 1980s, statistical NLP emerged to simplify parsing rules combined with rule-based approaches leveraged to enhance results (Nadkarni et al. (2011)).

Neural language modeling first appeared in the early 2000s, anticipating words by using their preceding words as a guide. For this reason, lookup tables and feed-forward neural networks were introduced (Bengio et al. (2000)). Later, this was expanded to multi-task learning using convolutional models, addressing named entity recognition (NER) and parts-of-speech (POS) in NLP (Brants (2000);Dalal et al. (2007);Khurana et al. (2023)).

In the 2010s, with the development of word embedding, NLP introduced neural networks, which accept different lengths of input for subsequent processing making this approach more prone to use around 2013. Thenceforth appeared the first notions fundamental to the later appearance of attention mechanisms around 2014 to increase the accuracy of language tasks by helping a model concentrate on keywords in a sentence,

allowing it to comprehend context and word relationships (Bahdanau et al. (2014)). The official development of the attention mechanisms as it is known now, was developed in 2017 by Vaswani et al. (2017).

Currently, Transformers, also created around 2017, continues to find use nowadays. Transformers are a specific neural network architecture that understands and generates human language, which makes them useful in tasks like translation, chatbots, etc. Their self-attention processes allow them to capture complex word associations and context (Vaswani et al. (2017)). Also, their adaptability has transformed NLP, which has undergone a revolution with the development of large language models such as BERT, GPT, and others.

Most advancements in neural network models are related to GPT-3, introduced in 2020. After the official launch of GPT-3.5, which was made possible for users to utilize it in 2022, the competitors have been incentivized to launch their own AI developments or even improve the existing contributions from researchers like Google's AI. Particularly, GPT has updated and improved its algorithm with a different approach than before, leading to GPT-4 in 2023 (Achiam et al. (2023)). The key components that urge great focus on improving NLP are the efficiency of algorithms in ethics and the reduction in bias (Kalyan (2023)).

Nowadays, using Large Language Models (LLM), often based on Transformers design, can efficiently improve metrics results. However, there are still many obstacles, namely processing time, meaning that there is still a large margin for the development of NLP (Naveed et al. (2023)). As we have seen so far, NLP has revolutionized the way we work, think, and act, bringing a greater focus to evolve this area. All of these historical development stages can be visualized in figure 2.1.

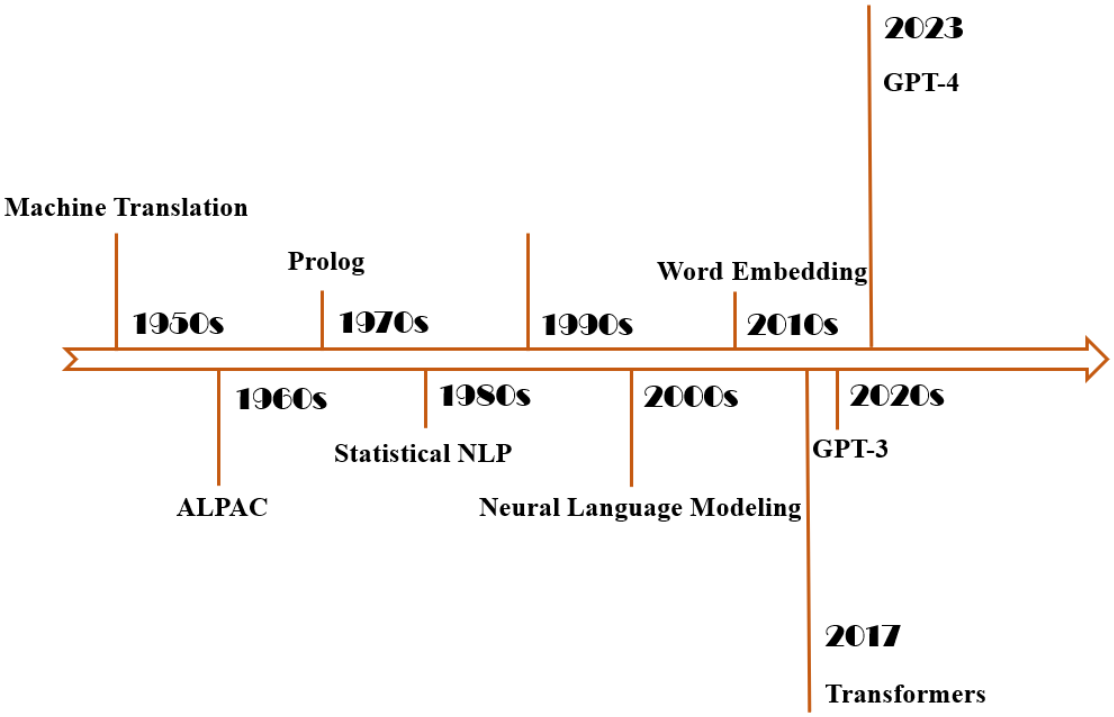

Figure 2.1: Chronology of Natural Language Processing

2.2 Natural Language Processing

NLP has become a vital area of research due to its extensive and complex evolution throughout time. This development has encouraged a number of subfields to concentrate on developing the disciplines of NLP naming Natural Language Generation (NLG) and Natural Language Understanding (NLU). Concerning NLG, machines generate spoken words or written text from a conversation or human utterances. During this process, they are expected to transform input data into meaningful sentences (Benmalek et al. (2019)). In terms of NLU, it entails taking every input expression and translating it into a machine way of understanding by taking out its metadata and analyze at all levels of linguistics like lexical, semantic (Feldman (1999)), etc. Also, machines are used to be developed to recognize and comprehend emotions and intentions in human communication, despite the need for further evolution. All in all, NLU is not limited to the exact meaning of words but rather is concerned with the entire cultural, historical, and contextual background that determines the use of a phrase or a word in a conversation, in order to better understand human thought. NLP can be viewed from different perspectives depending on the task at hand, but the main core is to process unstructured data into structure. For this research, it is important to note that both NLU and NLG will be key areas of focus, as they are expected to undertake significant development (Kang et al. (2020)).

2.2.1 Natural Language Generation

Sometimes, it is controversial to distinguish NLP tasks between NLU and NLG due to its ability to be characterized as being both. However, there is always a primary association of NLP-specific tasks with each of the NLP subfields. Next, the most common NLP applications will be associated with their principal subclass, in this case, NLG.

• ***GPT-3*** (Brown et al. (2020)) and structure-generated text – Independently of the input organization, after generating text, it should have introduction, development, and conclusion traducing in a human-readable structured output (Kacprzyk and Zadrozny (2010)). This is enhanced by applications like *GPT-3* and its successors named *GPT-4*.

• ***Narrativa*** (Skrodelis et al. (2023)) and content – *Narrativa* is an AI tool that serves the purpose of generating reports, news stories, or even summaries from text data. In order to have a good quality text generated, the way information is spread and the main objective message should be defined in terms of context and transmitted to the user (Reiter and Belz (2009)).

During NLG machine execution, the text generation faces challenges in terms of diversity, ethics, grammatical, tone, or style written issues to bring the best quality of the generated output to the user. Those challenges may imply the avoidance of the repetition of words, mitigating all types of discrimination like race, and the choice between a formal or an informal text (Sai et al. (2022)).

2.2.2 Natural Language Understanding

It is peculiar to identify which tasks are NLG or NLU, as was previously noted, however using the earlier methodology where common tasks will be considered with their main assignment, in this section, NLU pipelines are highlighted.

• **Named Entity Recognition (NER)** – From unstructured text, NER determines and categorizes entities found through understanding the context and semantics to extract organized information (Byrd and Ravin (1999)). NER may classify each word with one or more classifiers using data from the entire document (Chieu and Ng (2002)).

• **Part-of-speech tagging (POS tagging)** – As one of the fundamental steps used to analyze grammatical positioning, POS is essential to categorizing such as nouns, verbs, and adverbs to comprehend the context of a sentence (Jatav et al. (2017)).

• **Sentiment analysis** – identify the emotional part of the text in terms of tone and feelings that can be positive or negative in the case of binary labels or other types of categorization defined during the analysis (Arevalillo-Herráez et al. (2022)). These are only some of the fields where NLU needed to start evolving. In the development of them, it

is important to highlight some of the threats found involving cross-domain adaptability, lack of labeling data, variety in the way users express themselves, and unknowingly situations where there is no training data causing a bad generalization of specific events. In conclusion, ambiguity plays a significant role, being one of the most challenging aspects of NLU performance that requires several adjustments to accurately comprehend human language (Williams et al. (2017)).

2.3 NLP Analysis and Linguistics

The written text can be analyzed in different ways and NLP text is not an exception. The complexity that composes a sentence can be split into several grammatical aspects. Thus, the mechanism behind the correctness of a sentence may be different depending on each machinery pipeline. The different types of grammar analysis are shown below:

• **Discourse** – Instead of focusing on a sentence level, discourse analysis looks at the entire text, specifically examining its structure. Some features lead to analyses of language in context such as conducting anaphora analysis to identify repetitions. It also focuses on cohesion – for example, how pronouns and conjunctions are used and connected in a sentence – coherence, cataphora, connectors, and more (Sukthanker et al. (2020)).

• **Lexical** – Ensures that contextualized words or phrases are chosen to meet the requirements of the text generated. It is a commonly used POS tag, NER, and other similar subjects (Lee et al. (2019)).

• **Morphological** – Treats how words are formed compared to the root of the words in terms of prefixes and suffixes. These variations of words, usually imply the preprocessing steps called lemmatization and stemming (Martinez (2010)).

• **Phonetic** – Search for analyzing the sound associated with the context of the speech and it can be divided into three subcategories, phonemic, phonetic, and prosodic rules. First, phonemic analysis, where the existing fluctuation that occurs when pronouncing words together is captured through sound waves. Second, phonetic analysis, where the sound associated with each word is detected. Thrith, prosodic analysis states how words are intoned during the speech (Reshamwala et al. (2013)).

• **Pragmatic** – The success of interpretation lies in understanding how context influences language by attaching the speaker's intention. Pragmatic information is needed to solve anaphoric allusions through a comprehension of the speaker's objectives (Liddy (2001)).

• **Semantic** – The semantic perspective understands the meaning of sentences looking at different words and knowledge behind each context. Semantic can disambiguate polysemous words that imply considering terms that may have different meanings by definition, however only one meaning of the word will be considered (Huang et al. (2012)).

• **Syntactic** – Spot and collect the grammatical errors in order to correct them to achieve the desired structure of a sentence. Then, the parsing of sentences or words facilitates the analysis of syntactic hierarchies and relationships. The syntactic approach is required to check dependencies to understand their meaning in a sentence (Sag et al. (2002)).

Grammar is a subfield of linguistics that serves as the scientific study of language, focusing on understanding and correctly applying its rules. In addition to grammar, language has other particularities and specificities that are studied in sociolinguistics, neurolinguistics, anthropology, and historical linguistics, among others. These subjects aim to examine the social, cognitive, neurological, historical, cultural, and other relevant environments in which the target population resides (Castelle (2018)). It is essential to recognize that language has undergone continual adjustments over the years. What is considered correct and normative for us today may develop the language in a different direction tomorrow. This evolution can be driven either by the appearance of new words (e.g. new social media applications) or by changing the meaning of existing words or expressions. It is significant to remember that each of these levels adds to the comprehension of meaning and context (Peters et al. (2018)). In the following table 2.1, there is a brief example of each linguistic level described above.

Table 2.1: Seven linguistic levels based on SParC utterances data

Linguistic levels	Examples
Discourse	The utterance, "Which department has the most employees?" tends to be more concise to generate a query than "What is the number of employees in each department?". The reason for this is that, in the last utterance, it requires not only one department, but all departments, adding complexity to build the query.
Lexical	In the utterance, "Which department has the most employees?", the word "most" indicates a high quantity of employees, meaning the importance of words for the context solution.
Morphological	In the utterance, "Who is the head of this department?". The word "head" refers to a role or title instead of the common knowledge part of the body.
Phonetic	The way utterances are pronounced may lead to different meanings and purposes of the questions themselves. In text-to-SQL context, phonetic is not a major concern, due to written text rather than spoken words.
Pragmatic	In the utterance, "How many employees does each department have?" the focus is not only on employees and departments, but also seeks to measure the employee distribution within each department.
Semantic	In the utterance, "Tell me the name and position of the head of this department", where "head" refers to the leadership position inside the department. The word "head" could have more meanings depending on its context.
Syntactic	In the utterance, "How many employees does each department have?". It is an interrogation structure shown by "each department" as being the subject whose auxiliary verb is "does".

2.4 Statistics and Mathematics behind NLP text

A bag of words (BOW) builds a corpus-wide lexicon of distinct terms forming a vocabulary. The frequency of each word is what counts and every document models the vector of each word occurrence, then word sequence is irrelevant. Binary BOW may be one of the most used BOWs that only counts 1 if the word exists otherwise zero, but it is not the only approach that exists. In case BOW is not binary, then it counts how many times the words appear in a document or corpus. BOW elements are words that represent features alongside all words forming an extensive vocabulary and the computational input is the vectors withdrawn from documents or sentences (Tsai (2012)). The ordinary way to recognize the features of a text is tokens, which can be split into three [1] different groups specifically, word-based, character-based, and subword tokenization. Each one is explained in detail below:

[1] https://huggingface.co/learn/nlp-course/chapter2/4?fw=pt

• **Word-based tokenization** – Text is divided by words where each word is assigned with a number that generates an ID. Figure 2.2 demonstrates that the rules for those divisions are spaces and punctuation. The biggest disadvantage is that similar words will be considered as being different and unrelated words simply because one word is the plural of another (e.g. House versus houses) creating a huge vocabulary size.

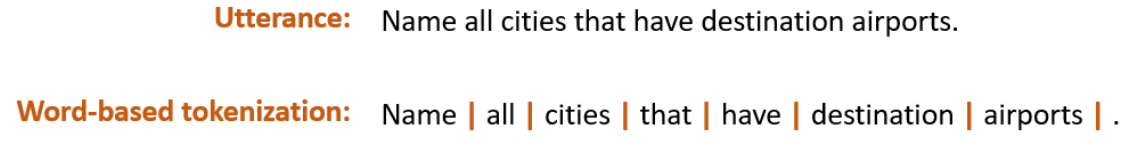

Figure 2.2: Word-based tokenization example

• **Character-based tokenization** – Text is broken through characters, not words as shown in figure 2.3. A character does not have a particular deep meaning by itself. As a consequence, the vocabulary is complete and reduced in size, in a way that all words can be written by the conjunction of several characters.

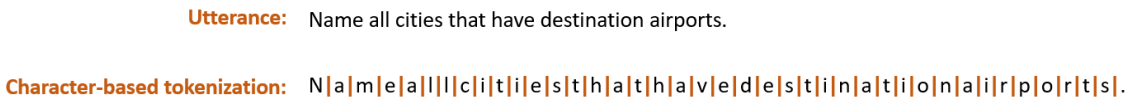

Figure 2.3: Character-based tokenization example

• **Subword tokenization** – This method mixes two previous approaches, word-based tokenization, and character-based tokenization in such a way that relevant words are not cracked since they appear frequently. In contrast, only rare words may be separate from their root form to have higher chances of appearing in the vocabulary as a standalone word as evidenced in figure 2.4.

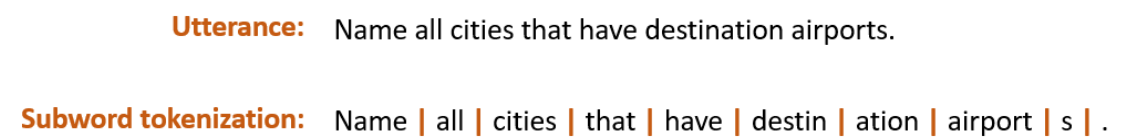

Figure 2.4: Subword tokenization example

The most common approach is the last one, subword tokenization since it considers the best of both approaches. However, tokenization brings issues with punctuation and compound words. To overcome that, the choice of a token may be central to help on that, it varies between type of task and type of language (e.g. in German, a good definition of compound words is crucial for interpreting the language). Thus, some of the approaches create a vast vocabulary that can lead to sparse vectors creating a massive BOW. This

may make it more likely that the model may overfit during training. This possible phenomenon, commonly known as the *curse of dimensionality* (Bromuri et al. (2014)), occurs when a model's performance is negatively impacted by having a high number of dimensions or features when compared to the number of training samples. One flaw of BOW is not considering similar words as sharing the same information (e.g. sea and ocean) (Collobert et al. (2011)). Whether the situation, commonly normalization should be done with dates, lowercasing all text, deleting stop words implying removing irrelevant words, like connectors.

Another path may concern the Term Frequency - Inverse Document Frequency (TF-IDF) which is commonly known for being better than BOW, because it takes into account the significance of the words in a text, and lessens the weight of often occurring words. Meanwhile, BOW only lists the number of times a word appears in a certain document. In contrast, TF is the proportion of a term's total number of occurrences in the document, represented as t, to its total number of appearances in a desired document, represented as d (Ponte and Croft (1998)). Moreover, by multiplying it with IDF, which counts all records most frequent in an entire corpus and reduces the chances of weights superabundant rare expressions. Then, an inverted logarithm is applied to the ratio between D and n_t. Here, the denominator, n_t, is the quantity of documents containing the word, and the numerator, D, is the total count of documents inside the corpus. This is illustrated in the following formula (Sparck Jones (1972); Qaiser and Ali (2018)).

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \log\left(\frac{|D|}{n_t}\right) \tag{2.1}$$

One particular TF-IDF is n-grams, focusing on n sequence of words, where n determines the number of followed words that appear most frequently together. The higher the n is, the higher the dimensionality, creating a bigger feature space. As seen before, this can lead to the *curse of dimensionality*, which can be a disadvantage. When modest n-grams are defined, they can capture contextual information, helping track the meaning, usually in case of an n greater than one.

2.5 Transfer Learning

For a human to perform a task, especially if it is complex, they must have prior knowledge in order to be able to perform the task successfully. This gave rise to the concept of Transfer Learning, which once again made it possible for the machine to resemble the human being. The idea behind the machine is to transfer knowledge from previously known tasks that are related to the topic in question (Hernandez et al. (2020)).

Pan and Yang (2009) have made a major contribution to this topic, through the trade-off between tasks and domains. As an example, we can identify how becoming a Spanish speaker, S, can be accelerated by knowing how to speak Portuguese, P. The two languages have similarities, making language transfer possible and fast. In this context, S refers to transfer knowledge, and P, means the source knowledge. The comparison happens between the different domains, learning different languages, $DS \neq DP$, where DS

refers to the Spanish domain and DP to the Portuguese domain. In case $DS = DP$ and $S = P$, then a traditional machine learning (ML) is presented (Kim et al. (2022)). Therefore, Pan and Yang (2009), state their goal to enhance a drawback of transfer learning, the algorithm becomes overwhelmed when the source and task are mistakenly identified as belonging to similar domains, leading to an inaccuracy. Then, a Heterogeneous Transfer Learning approach stands out as a secure result, where feature space has disparate characteristics in terms of tasks and a wide range of source domains (Pan and Yang (2009)).

2.5.1 Sequence-to-sequence models

Sequence-to-Sequence (Seq2seq) (Sutskever et al. (2014)) models are a class of neural networks that are commonly designed and referred to be used in NLP and are intended to convert sequences from one domain into another. Notice that it became more relevant with its use in machine translation, where the goal is to translate a phrase from one language to another, allowing for Seq2seq to gain significant focus.

Seq2seq built architectures were mostly constructed employing recurrent neural networks (RNNs) (Rumelhart et al. (1986)), including variations like Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber (1997)), prior to the introduction of Transformer models. These RNN-based Seq2seq models handled sequences well, because of their innate capacity to manage variable-size inputs and preserve information across time. Nevertheless, they were limited, especially when it came to managing long-term dependencies because of problems with expanding and vanishing gradients.

In order to overcome these problems, Seq2seq Transformers have appeared stating free use of candidate layer connected through self-attention mechanisms implying awareness by passing the testimony of context information (e.g. context text-to-SQL). In other words, attention methods were included, which enabled the model to concentrate on distinct segments of the input sequence at every stage of the decoding procedure. In this way, this led to a notable enhancement in the model's performance on tasks related to machine translation. Also, the area of sequence modeling saw a huge change and transformation when Transformer models were developed to solve the computational scalability and efficiency issues, that RNN-based Seq2seq models continued to have.

2.5.2 Transformers and Large Language Models

Looking at transfer learning as a whole, neural networks take the big picture due to their ability to create pre-trained models and carry information from one task conveniently used into another, even if there is a lack of labeled data. Innovation ensured that neural networks were modified with Transformers' new design that focused on self-attention mechanisms (Shaw et al. (2018)). The following topics demonstrate the most common parts of a Transformer model's architecture, which are made of two main components categorized as follows:

• **Encoder** – Converts text input into numerical illustrations called features through a self-attention approach. The encoder is associated with an input whose main goal is to find its context followed by the creation of a well-defined vector (Raganato and Tiedemann (2018)). Only encoders perform tasks like NER and sentiment analysis.

• **Decoder** – Experience similar functions compared to the encoder, meaning the acceptance of text as input through a masked self-attention approach that is a specific type of self-attention mechanism (Song et al. (2018)). Usually, decoders are used in an autoregressive way, in other words, the output generated from each step is kept to the following input for the subsequent step. Only decoders focus on tasks such as text generation.

These two approaches can be combined into only one solid concept, encoder-decoder. Also known as sequence-to-sequence (Seq2seq) Transformers. In order to provide a better understanding of the Seq2seq framework (Lu et al. (2021)), now analyzing the computational perspective. Due to the fact that machines process information over numbers and mathematical formulas, one of the first steps is to traduce input text into numbers.

In the encoding stage, those numbers keep information about each word and phrase. Consequently, they get details from the sequence. However, with the attention mechanism, the layers only give specific attention to certain words due to their meaningful context. These models are sometimes referred to as having bidirectional attention (Sun et al. (2019)), because they accurately predict words in the middle of a sequence. Most encoders, in traditional Transformers, use a self-attention mechanism that processes input sequences $(x_1, \ldots, x_n)$, creating representations for the decoder. These are then passed to generate onto output sequences $(y_1, \ldots, y_n)$ of the same size, knowing that all items are independently computed. Going in-depth, from the input, it is extracted three vectors, query (Q), key (K), and value (V). Only the first two will be multiplied to generate the attention score, knowing that d_k is the dimensionality of both vectors, shall look at the following formula (Vaswani et al. (2017);Vuckovic et al. (2020)):

$$a_{ij} = \text{softmax}\left(\frac{Q_i \times K_j}{\sqrt{d_k}}\right) \tag{2.2}$$

Softmax normalizes i and j positions of attention scores that bring the relevance of the different positionings of words in a sentence. Notice, that this normalization takes into consideration that it is only applied in the last dimension of each layer. To achieve the final output, one more calculation needed to be computed, as such the total sum of the weighted vector value of the words is mathematically represented as (Vaswani et al. (2017);Wang et al. (2020)):

$$y_i = \sum_{j=1}^{n} a_{ij} \times V_j \tag{2.3}$$

Upon encoder context vector output that now serves the decoder's input, a masked self-attention is commonly adopted, allowing the model to look only at the left of the words without a look at the right context of them. In other words, the model takes in consideration the past words without a glance at further words in a sequence to avoid being affected by wrong sequential order to generate the output (Dai et al. (2020)). This

is achieved by masking the future words to let sentences be analyzed chronologically, finally, the prediction is fulfilled to reach their target point.

This can be a guide for future model decisions by defining which kind of task to perform and the architecture model approach to choose. Transformers usually are the base for many Large Language Models (LLM) due to their ability to capture contextual long-term dependencies that come from the pre-training done in huge amounts of data. Frequently, LLM are pre-trained LLM and not built from scratch as a result of less time consumed in data collection for the required training. Special focus on high-demand time and computer resources is needed in case of no pre-train (Chen et al. (2021)). Currently, pre-trained LLM has reached many state-of-art in different benchmarks being a key factor in most NLP applications. In the following figure 2.5, there is an encoder-decoder architecture model, which offers an explicit visualization of the model's structure.

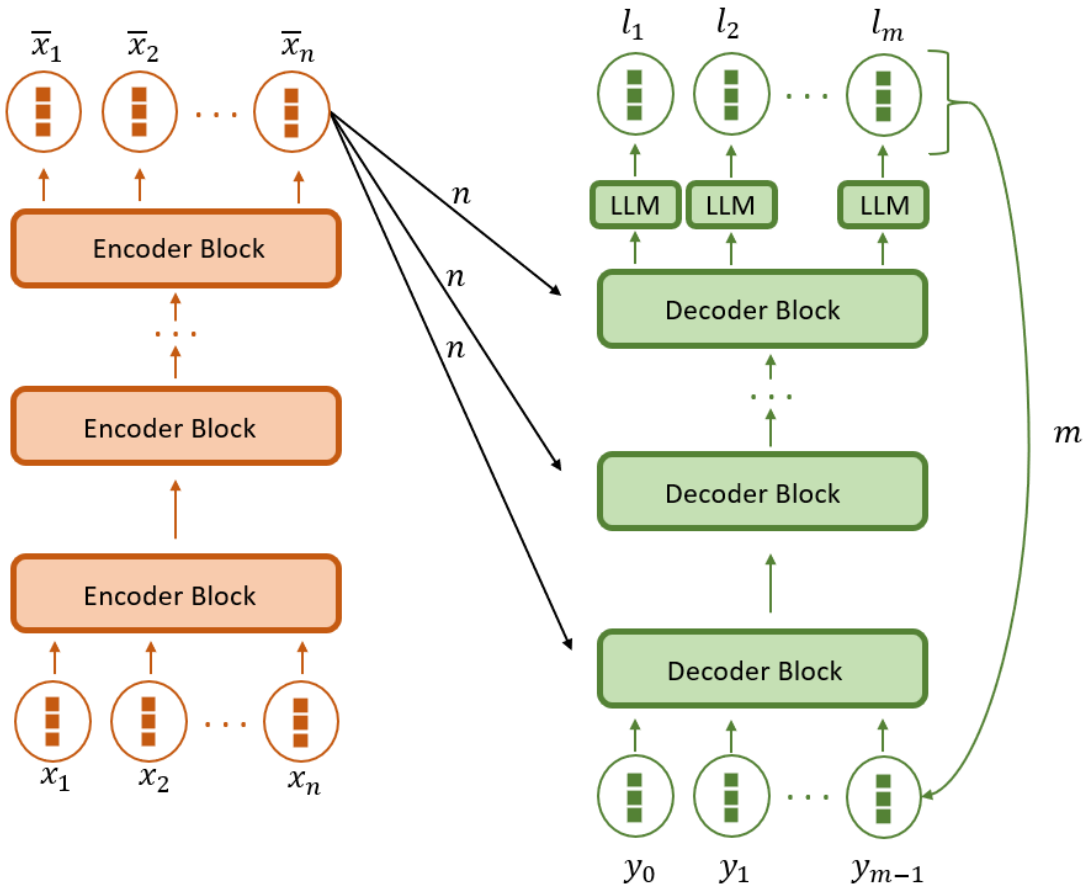

Figure 2.5: Encoder-decoder architecture model

Chapter 3

Related Work

3.1 Text-to-SQL Metrics

SParC benchmark (Yu et al. (2019b)) provides two different types of evaluation data to be compared, gold and predicted files [1] data, that determines whether a projected SQL query and a gold query exactly match, this kind of metric is known as Exact Set Match or Query Match Accuracy. In more detail, gold is the ground truth collection of accurate values or results that are used to assess how well a model serves as a reference to resemble during the validation or testing phase. The tables mentioned in the "FROM" clause of both predicted and gold queries that are being compared are considered as having exact match when there is a precise correspondence between the tables. This analogy is not only made at the "FROM" clause level, but also at other query components. Through all of these calculations, the F1 score strikes equity between precision and recall to compute on exact set matching (Yu et al. (2018b)).

Specifically, Exact Set Match can take into account value results from queries, although due to the complexity of text-to-SQL, Exact Set Match without values appeared to facilitate, since it is difficult to predict the same value, particularly when Natural Language Query (NLQ) storage representation varies from that current database (DB) storage. This metric is typically found in *Spider* benchmark researches (Yu et al. (2018b); Katsogiannis-Meimarakis and Koutrika (2023)). Furthermore, *Spider* gauges Exact String Match which is neither more nor less than literal string comparison disregarding ordering or syntactic variations that might nevertheless reflect the same query when examining if the SQL query meets the ground truth entirely. This can lead to false negative predictions (Zhong et al. (2020a)) when the projected inquiry does not match the ground truth. As a visual example, figure 3.1 presents a semantic text-to-SQL parsing gold link through schema linking.

[1] https://github.com/taoyds/sparc/tree/master/evaluation_examples

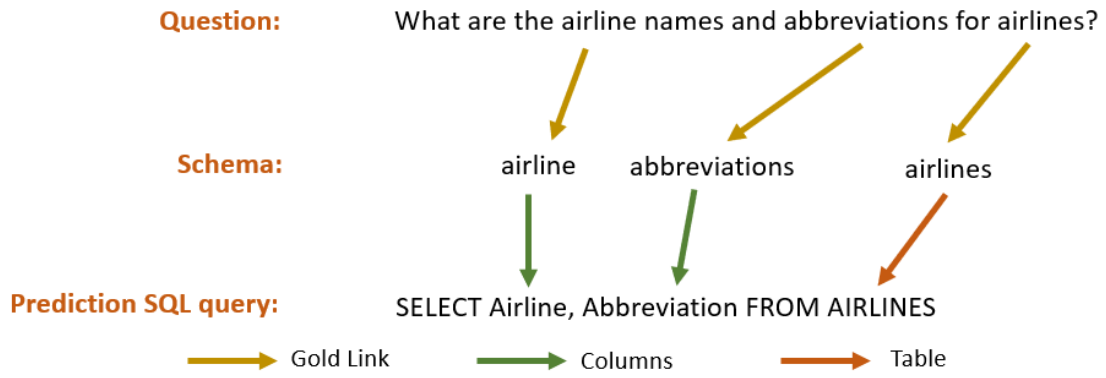

Figure 3.1: Semantic text-to-SQL parsing gold link through schema linking

As a Sub-tree element matching, a Partial Component Match F1 (PCMF1) arose as a solution to overcome Exact Set Match drawbacks, it computes F1 scores for each sub-tree component, such as the "SELECT", "FROM", and "WHERE" clauses, grounded in recall and precision of each predicted columns. PCMF1 deviates from the exact match metric. It awards a score approaching a value of 1 when extensive queries are accurate in some segments, despite casual errors, such as mistakes within the "WHERE" clause. This approach prevents a score of 0 for portions of the query that are right, allowing for a complex assessment of the system's performance. This procedure was tested in the *SEDE* (Hazoom et al. (2021)) dataset. To clarify the PCMF1, it is important to assess SQL queries as being formed by categories, or highly known as SQL components, represented as C = {SELECT, TOP, FROM, WHERE, GROUP BY, HAVING, ORDER BY}, each query can be denoted as q, connoted with predicted query q_p and a gold query q_g. A set of elements in a category is imperative, denoted as $s_c(q_p)$ or $s_c(q_g)$ according to prediction or gold set knowing $c \in C$. As a math's illustration:

$$\text{PCM-F1}(q_p, q_g) = \frac{1}{|C|} \sum_{c \in C} F1 \times (s_c(q_p), s_c(q_g)) \tag{3.1}$$

Regarding Execution Accuracy (EX) or Query Accuracy calculates how a query prediction seems to be accurate through the comparison of the ground truth. In the end, the output can be equal and present false positive outcomes when the queries are semantically distinct, and the output presents missing values or similar results values from different columns or tables (Parthasarathi et al. (2023)). From another work tested on *SParC*, *Spider*, and *CoSQL* benchmarks, a particular meticulous type of EX is Fuzz-test accuracy (FX) that was applied to the process of repeatedly running queries on a random or arbitrary set of DB entries in order to minimize the possibility of false positives (Zhong et al. (2020b)).

Exact Match (EM) and EX are the official evaluation methods in the *SParC* benchmark, although the other metrics are also very important to calculate depending on the project need, such as Interaction-level Exact Match (IEM) and interaction-level Execution accuracy (IEX) (Qi et al. (2022)). While EM and EX look forward to individual

question-answer, interaction-level EM and IEX pay attention to the entire dialog operating at a more robust-grained level instead of the fine-grained level in EM and EX.

3.1.1 Benchmarks of text-to-SQL

Currently, in the text-to-SQL task there is different data available online to test and grow this area showing distinct cross-domain data available such as *Spider* (Yu et al. (2018b)), *SParC* (Yu et al. (2019b)), and *CoSQL* (Yu et al. (2019a)) or sometimes just one domain like *WikiSQL* (Zhong et al. (2017)). As a starting point, it is prudent to look into the current first-place leaderboard available content and check the characteristics of the data and the tools used to achieve good score results.

In *Spider*, in the first position of Execution with Values, there is DAIL-SQL + GPT-4 + Self-Consistency (Gao et al. (2023)) that explores three key aspects related to supervised fine-tuning (SFT), question representation, and finally in-context learning. In this specific research, GPT-4 was the LLM chosen knowing that the open source LLMs prove better results when adding parameters. Also, supervised fine-tuning was stated as mandatory due to its high potential. Spider benchmark was considered poor because considering a possible increase in the DB may affect the effectiveness of the processes developed. In-context mainly concentrates on the target SQL answer or question observing a significant reduction of learning capability afterwards fine-tuning. Now, the Exact Set Match without Values distinguishes the Graphix-3B + PICARD from Li et al. (2023b) that states a hybrid approach with a complete T5 model as base, plus graph-aware layers, turning it efficient in terms of contextual encoding. Also, multi-network thinking, traducing in a superior methodology, when compared to applying only T5-based model version.

In terms of data, *Spider* shows 138 distinct domains, with 5693 SQL answer queries to 10181 utterances, while *SParC* is richer in terms of queries complexity and contextualized information, being closer to the real-world environment, having 4298 sequence utterances, plus more than 12000 unique utterances. In the leaderboard of *SParC* concerning Execution with Values, distinguished RASAT + PICARD created by Qi et al. (2022), where a Transformer Seq2seq model was implemented while using the pre-trained T5 model parameters to considering schema encoding and schema linking, achieving a notable performance by applying this methodology. In this research, it demonstrates self-attention with the encoder and brand-new parameters to the model keeping the pre-trained weights. In the leaderboard of *SParC* of Exact Set Match without Values specifically in the first position, there is a STAR research project developed by Cai et al. (2022) highlighting two distinct components, namely schema state tracking (SST) and utterance dependency tracking (UDT). The first main goal is to employ SST to investigate the schema in terms of context dependencies regarding SQL queries. In this way with all the interactions, each schema is treated with SST forecasts and fits the values adjusted to each schema slot. The second objective utilizes weighted contrastive learning. This approach works by applying the recognition and maintenance of equivalent semantic meaning of natural language questions. At the same time, it separates questions when they use similar semantic terms, but in different contexts throughout the all dialog.

Another benchmark highly related to *SParC* and Spider is *CoSQL* (Yu et al. (2019a)), which is a conversational version of previously mentioned datasets. This extension also has the STAR study from Cai et al. (2022) as their first ranked Exact Set Match without Values, achieving a Question Match score of 57.8 and Interaction Match score of 28.2, which is way lower when the same study and methods are applied to *SParC*, demonstrating a clear need to evolve these techniques. The same happens in the Execution with Values with RASAT + PICARD achieving worst results when compared to *SParC* and Spider.

Since *SParC* is a benchmark that encompasses several domains, more focus was given to cross-domain research studies beyond *SParC* like Spider and *CoSQL*. In the following tables, table 3.1 and table 3.2, consider the leaderboard scores of the respective models.

Table 3.1: Execution with values in text-to-SQL benchmarks

Benchmarks	Model	Authors	Test	Question Match	Interaction Match
SParC	RASAT + PICARD	SJTU LUMIA & Netmind.AI (Qi et al. (2022))		74.0	52.6
	TreeSQL V2 + BERT	Anonymous		48.5	21.6
	GAZP + BERT	University of Washington & Facebook AI Research (Zhong et al. (2020b)))		44.6	19.7
CoSQL	RASAT + PICARD	SJTU LUMIA & Netmind.AI (Qi et al. (2022))		66.3	37.4
	GAZP + BERT	University of Washington & Facebook AI Research (Zhong et al. (2020b))		35.9	8.4
Spider	DAIL-SQL + GPT-4 + +Self-Consistency	Alibaba Group (Gao et al. (2023))	86.6		
	DAIL-SQL + GPT-4	Alibaba Group (Gao et al. (2023))	86.2		

Table 3.2: Exact set match without values in text-to-SQL benchmarks

Benchmarks	Model	Authors	Test	Question Match	Interaction Match
SParC	STAR	Alibaba DAMO & SIAT (Cai et al. (2022))		67.4	46.6
	RASAT + PICARD	SJTU LUMIA & Netmind.AI (Qi et al. (2022))		67.7	45.2
	CQR-SQL	Tencent Cloud Xiaowei (Xiao et al. (2022))		68.2	44.4
CoSQL	STAR	Alibaba DAMO & SIAT (Cai et al. (2022))		57.8	28.2
	CQR-SQL	Tencent Cloud Xiaowei (Xiao et al. (2022))		58.3	27.4
	RASAT + PICARD	SJTU LUMIA & Netmind.AI (Qi et al. (2022))		55.7	26.5
Spider	Graphix-3B + PICARD	Alibaba DAMO & HKU STAR & SIAT (Li et al. (2023b))	74.0		
	CatSQL + GraPPa	Anonymous	73.9		
	SHiP + PICARD	AWS AI Labs (Zhao et al. (2022))	73.1		

3.2 Text-to-SQL Limitations

From previous text-to-SQL research and conversational text-to-SQL research developed until now, several limitations have been found across all benchmarks. Those drawbacks are key components to focus on future investigations in the area. Subsequently, there is a closer perspective of each major concern. Starting from serialization input passing to output decoding, ambiguity, and more. There is a clear message behind exploring the existing drawbacks of text-to-SQL, which involves knowing the successive progress and stating where we are today to develop cutting-edge processes tomorrow. The subsequent sections describe the most common limitations.

3.2.1 Input Encoding

Input data has different treatments according to their size. Although with *Spider* benchmark there is no concern about input size serialization. However, in most real DB,

where pre-trained LLM and serialization techniques are used together to encode input data, this encoding is typically limited to around 512 tokens. This limitation, as noted by Katsogiannis-Meimarakis and Koutrika (2023), poses difficulties in coping with wider DB schemas. Serialized inputs are usually structured at one glance onto one single sequence to further encode serving as an input for the neural encoder, since the sequence is unique, the context is not lost for pre-trained LLM. Pre-trained LLM has a restricted input size unable the serialization of the schema that would produce in a long sequence. In the same perspective, if the schema is split, then it may result in a bottleneck in the performance of the schema encoder. As a solution, Graph Neural Network (GNN) proved to be more effective in managing bigger DB schemas. Nevertheless, as the DB grows this method tends to decrease its effectiveness.

Most of the approaches use tabular data as their main source of DB content. A parallelism of both tabular and textual data that shares the same trouble regarding the development of a strong input encoding method that can cope with wide schemas. Within the domain of tabular data sources, pre-training methods entail the transformation of two-dimensional data towards one-dimensional data structured in sequences, which are then introduced onto language models. Serialization was used to compress tables' rows, one by one, which is demonstrated in *TABLEFORMER* (Nassar et al. (2022)), *MATE* (Eisenschlos et al. (2021)), and *TAPAS* (Herzig et al. (2020)) models as mentioned in Qin et al. (2022). Text-to-SQL needs to be prepared to handle all kinds of input data to avoid possible problems at the starting point.

Although representing input as a graph reduces the disappearance of information and supports different kinds of extra inputs, however, this resides in the complexity of executing a graph within a neural network when contrasted to simply executing a sequence. This easily turns into a challenge due to GNN's ability to turn all the execution way more complex, this meant this process to not be chosen by many projects for a long time. To break this trend, in the successful RAT-SQL work, a Transformer was implemented with GNN achieving promising combination directing for future studies.

3.2.2 Output Decoding

The sequence-based method is considered by many as the primary and basic approach developed, which was introduced in Seq2SQL, where a decoder technique implemented a deep learning approach to produce a SQL sequence of schema and tokens. It did not take long for its disadvantages to appear regarding any precautions to prevent syntactical wrong SQL code generated without being spotted and avoided, immediately making it easy to create names for tables and columns that do not even match or exist in the original DB due to the blindness of SQL grammar code. Recent studies show renewable progress with pre-trained LLM plus *PICARD* (Scholak et al. (2021)), this last one avoids misspelling of SQL code to impede major errors.

After those conclusions, other researchers tried to overcome *PICARD* as such urge *CatSQL* model (Fu et al. (2023)). *CatSQL* is a sketch-based architecture that does not imply common words as keywords requiring less time to run and having better accuracy in getting the first position in the leaderboard of Spider at that time. In general, sketch-based

methods find a smooth way to classify activities to well retrieve predicted SQL query table columns as an example. Nevertheless, until reach that point, mockups for every type of SQL query are tough making it hard to generalize. Thus, all interactions inside the neural networks become a final huge *"black-box"*. A *"black-box"* in this case means not knowing exactly what is behind all decisions made throughout the entire neural network connection processed. Meanwhile, with *CatSQL* the results also overcome *SmBoP* (Rubin and Berant (2020)) and *RYANSQL* (Choi et al. (2021)) sketch-based models.

Grammar-based decoders are particularly complicated to accomplish because they need a complete grammar dictionary of rules to address every potential SQL query. Even though grammar-based, when compared to Seq2seq, often has better syntactical SQL correctness and proper order tokens generated. Taking closer attention to the most popular works using the grammar-based approach as their main SQL queries producer are RAT-SQL (Wang et al. (2019)), *SyntaxSQLNet* (Yu et al. (2018a)), *IRNet* (Jha et al. (2019)), etc. (Katsogiannis-Meimarakis and Koutrika (2021)). Knowing that the best final technique for output decoding has a lot of space to grow, meaning no certainty about the best approach to choose. Lately, there has been substantial progress in the development of sequence-based decoders improving their notable capabilities and performance (Katsogiannis-Meimarakis and Koutrika (2023)).

3.2.3 Ambiguity Text-to-SQL

There are several types of ambiguous natural language text that can be analyzed in terms of grammatical issues, and others, with an attempt to be resolved in different ways. From some literature underlines decoding algorithms as producing every possible interpretation for future user clarification of the ambiguity. One of the main issues found in managing ambiguity is standard beam search techniques and their modifications. A brand-new decoding method entitled LogicalBeam (Bhaskar et al. (2023)) was created to overcome such limitations regarding inadequate token-level variety that usually considers SQL queries as strings. To provide an overview of LogicalBeam that combines limited fill-in to move across the SQL logic domain space with a plane-based model structure generated. Instead of traditional beam-search that concentrates only on schema names, should be created plans to help diversify across model structured the multi-variations of values during fill-in. This procedure was tested in Kaggle DBQA and *Spider* reaching the fifth-best results in terms of Execution Accuracy (EX) and Exact Match (EM).

Moreover, another way of dealing with ambiguity that may be a more convenient method for the users arises when the users' questions are not answered in SQL. They receive a message to rewrite the question, becoming the users actively overseen by direct involvement to improve the communication linkage between users and machines (Yu et al. (2019a)). Alternatively, simply inform them that those answers to those questions simply do not exist, and there will be no answer at all, this is one of the methods defended in *CoSQL* projects. In case the users do not find a way to make the question properly understood by the machine, this can lead to users' time-consuming and a requirement to build a guideline for them to know which kind of questions they should do to fit in the right format of a system to further understand and generate SQL responses.

Paraphrasing is also an ambiguous language problem that can use *WordNet* (Gkini et al. (2021)) as a tool to provide equivalent words when the machine spots an unknown sentence by changing from the unrecognized term to a synonym that might be identified in a DB content by the machine, meaning a final SQL answer given. However, there are many related issues with ambiguity, those shown until this point are just some of the solutions found and tested in common studies. It is also important to mention that context-dependent ambiguity can be defined as whose word meaning can be attributed based on query context, set of user main objectives, and domain studied because the same word can have different meanings (e.g. cell from biology context or cell from an excel unit within a spreadsheet).

It should be noted that many of the datasets available to test algorithms, such as *Spider* for example, are still very far from the DB used in a real context, presenting much lower levels of complexity when compared to the real-world. This means that even if the results of the current studies are excellent, it does not mean that text-to-SQL has been found as a wide solution to all DB. It is impossible not to mention the cross-domain adaptability problem faced by applying the same techniques to different benchmarks resulting in different statistical value outcomes. All difficulties are opportunities for future improvement resulting in the evolution of text-to-SQL and conversational text-to-SQL.

3.2.4 Tackle Schema Linking

The most recent state-of-the-art works defend the linkage of Natural Language Query (NLQ) plus Databases to the NLP through schema. There are two different types of schema linking, candidate discovery, and candidate matching, knowing that the broad area is taxonomy. It is possible to categorize in more detail as a column link or table link when the query matches the column or table and finally, a value link, when a query meets the value of a column (Katsogiannis-Meimarakis and Koutrika (2023)).

Comparing different approaches, RASAT + PICARD from the *SParC* benchmark, designed an implementation of a T5 model that leverages the encoder part of the self-attention modules by adding new parameters without losing the pre-trained weights (Qi et al. (2022)). *PICARD* finds acceptable output sequences by removing tokens that are not allowed at every stage of decoding (Scholak et al. (2021)). In this case, schema linking is highly affected by the RASAT models' architecture that englobes schema linking, encoder, and decoder. Specifically, n-gram was used to match and locate occurrences to overcome difficulties in relation detection, which can be analyzed in two different ways, exact and partial matches to avoid the impact of wrong matches.

In a different benchmark, *Spider*, RAT-SQL approach develops schema linking focusing on matching names and values. Regarding match names, the Relation-Aware Transformer (RAT) framework overcomes Transformer model barriers related to the lack of textual matches by using n-grams to encode names. Thus, a value match from the query is linked to the respective columns by the *RAT* approach using previous knowledge, precisely, the database information itself (Wang et al. (2019)). Another benchmark, *WikiSQL*, defends removing noise by applying shuffle and erosion. Erosion modifies connected schema entities in SQL queries by arbitrarily changing, increasing, and removing

columns from the schema input. Shuffle is a technique for monolingual self-supervision that reorders entities and values in SQL or Natural Language (NL) at random to help the model retrieve original text from noisy input (Xu et al. (2021)).

In all benchmarks, their schema linking approach achieved state-of-the-art results. Comparing one another, not all benchmarks face schema linking problems with equal significance, as the nature of the data varies across different instances. Compared to *Spider*, encoding the schema in *WikiSQL* is not as difficult because multi-table relations are not involved in *WikiSQL*. Meanwhile, schema linking is necessary for both benchmark activities, Spider offers more adversity being more expressive with NL and less constrained with SQL syntax. In turn, RASAT proved to be efficient on *Spider* surpassing its state-of-art by achieving high scores in fine-tuning execution accuracy of 75.5, showcasing a 0.4 solid advancement over text-to-SQL tasks performance (Qi et al. (2022)). A more visual way to perceive the same project, RASAT + PICARD, achieving different results according to each benchmark is shown in the table 3.3.

Table 3.3: Question and Interaction Match and Test scores in text-to-SQL benchmarks regarding RASAT + PICARD

Benchmark	**Model**	**Authors**	**Execution w/ Values**		**Exact Set Match w/o Values**	
			Q.M.	**I.M.**	**Q.M.**	**I.M.**
SParC	RASAT + PICARD	SJTU LUMIA & Netmind.AI (Qi et al. (2022))	74.0	52.6	67.7	45.2
CoSQL	RASAT + PICARD	SJTU LUMIA & Netmind.AI (Qi et al. (2022))	66.3	37.4	55.7	26.5
			Test		**Test**	
Spider	RASAT + PICARD	SJTU LUMIA & Netmind.AI (Qi et al. (2022))	75.5		70.9	

Chapter 4

Methodology

4.1 Overview

To initially examine the methodology, we have to take into account the overview sections that build the whole workflow of this project's methodology. The first section focuses on the preprocessing data from the *SParC* benchmark, focusing on schema linking and semantic parsing through tree dependency parsing. This is followed by the second section, pursued by the modeling architecture section, mainly focusing on the training with T5-base model on *SParC* data against the inference using the T5-base model previously validated with *WikiSQL* data, followed by a fine-tuning model to generate SQL using *SParC* data. Finally, the last section gives intuit to the different types of evaluation of the final algorithms' performance. The figure 4.1 illustrates a detailed overview of the previously mentioned sections, including a representation of each step's methodology workflow.

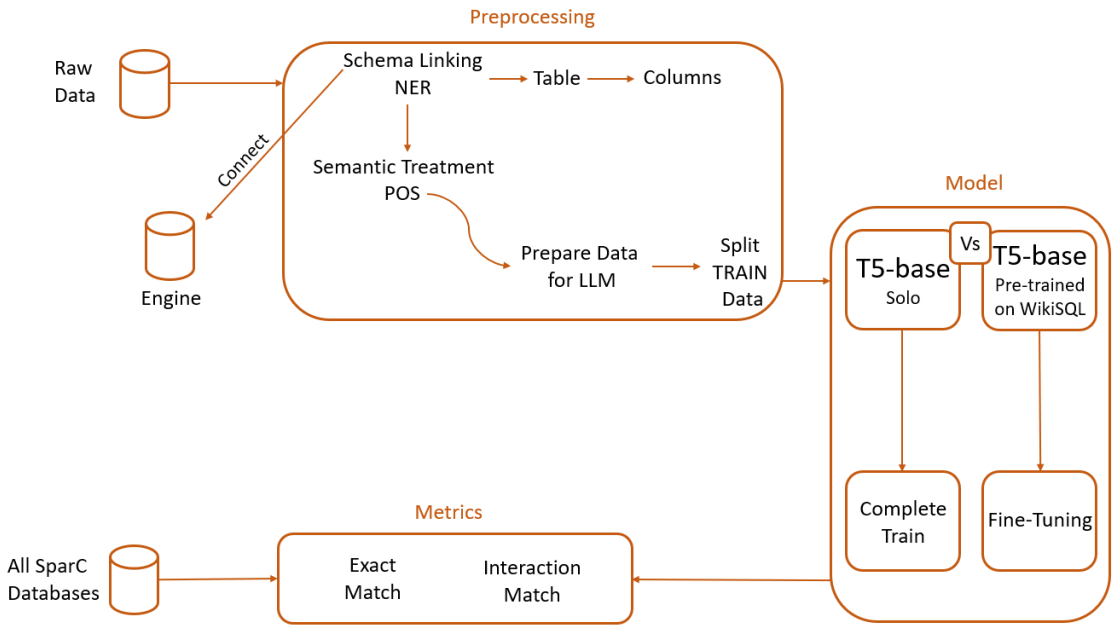

Figure 4.1: Methodology Workflow

4.2 Preprocessing

After NLP tokenization, some preprocessing steps are required to convert NL queries into an organized format suitable for feeding into generated SQL query models. For example NER, schema linking and semantic treatment, namely semantic parsing and POS, are key core components in this section.

NER particularly gathers and categorizes distinctive informational fragments of NL queries to map SQL query components that compose each SQL statement. Text-to-SQL NER can be analyzed in different extents. First, the entity may refer to table names for later usage inside the "FROM" clause, which is SQL generated to get the required data, while ensuring the relationship between tables with join clauses. Second, the entity to be identified is column names, later used in the "SELECT" statement, also it is required the identification in utterances questions to spot the aggregation words that correspond to SQL functions. In case necessary, some filter entities would be recognized and applied within condition values inside what is generated by the "WHERE" clause, including quantifiers, logical operators, temporal entities, or even geographical entities. Usually, this is followed by aggregate conditions like "GROUP BY" identified in grouping values, for instance, total sales. In table 4.1, there is a prototype example of the constitution of the following query to their NER, "*SELECT advisor.name, SUM(advisor.salary) AS total_salary FROM advisor JOIN department ON advisor.department_id = department.id WHERE department.name = 'History' GROUP BY advisor.name*".

Table 4.1: Query decomposition to demonstrate entity type

SQL Components	**Entity Type**	**Examples**
SELECT	Column Names	`SELECT name, salary`
SELECT	SUM, COUNT, AVG	Aggregation Function `SELECT name, SUM(salary)`
FROM	Table Names	`FROM advisor`
JOIN	Join Tables	`JOIN department ON advisor.department_id = department.id`
WHERE	Filter	`WHERE department_name`
WHERE	GREATER THAN, EQUALITY, BETWEEN	Logical operators and temporal entities `WHERE department_name= "History"`
GROUP BY	Grouping values	`GROUP BY name`

Parsing core functionality uses *stanza* pipeline to entail the extraction of information from each word in all utterances by doing POS, and dependency parsing relation. POS

tagging focuses on the grammar structure of the questions, locating and categorizing each word meaning auxiliary verb, noun, main verb, adjective, and others, with its appropriate part of the utterance text in NL to better understand and generate a query's structure correctly. Knowing the grammar can help spot the different meanings of words depending on their usage in an utterance to generate correct SQL queries through POS recognition. For example, a noun is an object that can refer to the domain, meaning the name of the table or column in a DB. POS will be executed with the *stanza* library (Qi et al. (2020)).

Dependency parsing focuses on the grammar relation between words by defining a "root" word and its corresponding interrelations words that are dependent on it. The grammar classification forms for these relations are syntactic link dependencies to the "root" word. Usually, the "root" word is the verb or predicate, that contains the main action in the question, being selected just one word. Then, in every word relation, except for the "root" word, there is a "head" of the dependency parsing tree. But, it is possible that the "root" can be the "head" word of the relation established between words. In table 4.2, there is an utterance example regarding POS and dependency parsing, "What is the number of employees in each department?", where it is possible to check each word's grammar regarding POS and dependency parsing.

Table 4.2: Examples of POS and dependency parsing

Type of Preprocessing	**Utterance**	**Examples**		
POS	What is the number of employees in each department?	**Utterance**	**Grammar**	
		What	WH-question	
		is	Verb	
		the	Determiner	
		number	Noun	
		of	Preposition	
		employees	Noun	
		in	Preposition	
		each	Determiner	
		department	Noun	
		?	Punctuation	
Dependency parsing	What is the number of employees in each department?	**Utterance**	**Head**	**Relation**
		What	is	nsubj
		is	ROOT	root
		the	number	det
		number	is	attr
		of	employees	case
		employees	number	nmod
		in	department	case
		each	department	det
		department	employees	nmod
		?	is	punct

Overall, the connection with the database is established through schema linking, which involves encoding the schema to link database elements with data utterances. This facilitates connections across different domains within the database, thereby reducing ambiguity. The algorithm structure is later explained in detail in section 5.1. Ultimately, more accurate and pertinent SQL queries can be generated as a result of the increased ability to process and comprehend queries. At the end of all of these changes, it is expected that data gain different dimensionalities that need to be adapted to be inserted inside the model.

4.3 Modeling

This modeling section explains the two main model developments, one with T5-base model trained on *SParC* data and a second model with T5-base model inferred with *WikiSQL* further finetuned with *SParC* data, focusing on conversational text-to-SQL problems that leverage the NLU of the utterances and, finally, the NLG of the SQL queries from those questions.

Regarding the first model, T5-base is the downstream model from the hugging face transformers developed by Google. In this ablation study, T5-base model does the complete train on top of *SParC* data. Solo training benchmark data can help to have a base reference when comparing to another model. That is only possible due to the preservation of preprocessing and equal evaluation systems. The solo training model benefits from specific domains and tasks regarding *SParC* data avoiding interference of third-party data that can be irrelevant. T5-base model incorporates a self-attention mechanism to be able for the model to deal with different lengths of input sequences by knowing that only the important tokens will have meaningful consideration by the model regarding relevancy of information and context. In figure 4.2, it is possible to see the general workflow of the ablation study on T5-base model approaches.

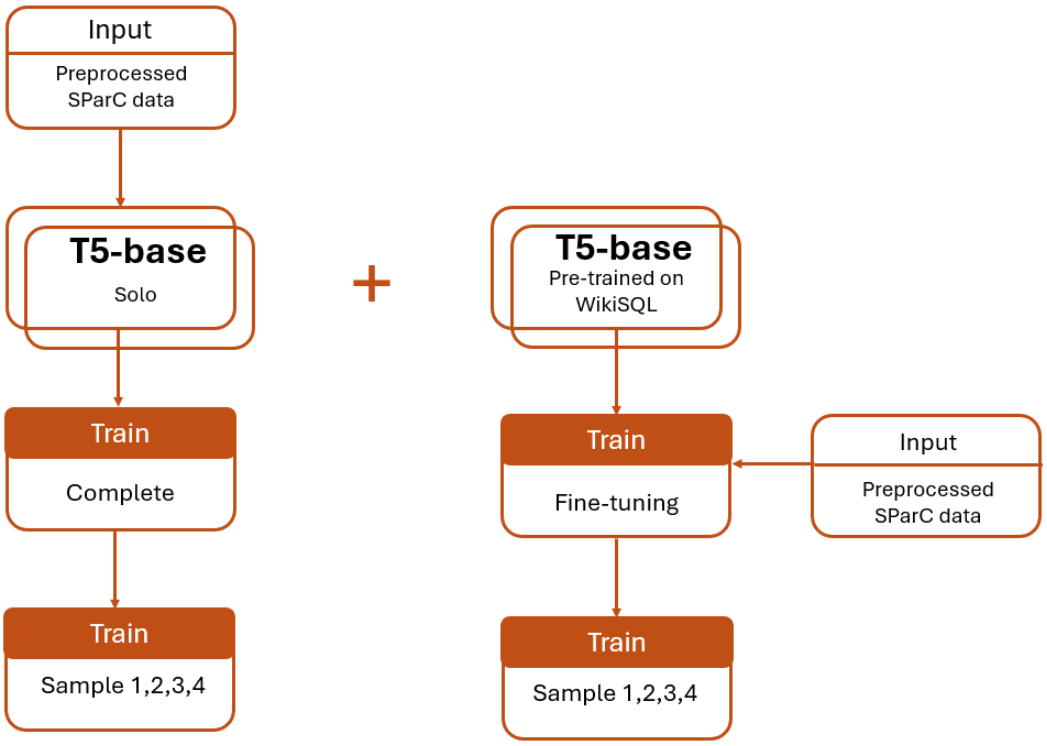

Figure 4.2: Ablation Study on T5-Base Models

Regarding the second model, a pre-trained model already fine-tuned with *WikiSQL* data was inferred first, due to *WikiSQL* abilities to generate SQL queries, specially trained on a T5-base model to create simple queries based on Wikipedia data. Further, adapted to *SParC* data through fine-tuning, on this occasion to generate simple, medium, and complex query tasks to this new unseen data for the model. After the implementation of the inferred model, fine-tuning is required to focus on conversational text-to-SQL specific problems. This method's target is to enhance accuracy by adapting the model to learn about different specificities of vocabulary domains and schema connections, from entities to the database columns, resulting in improved SQL query syntax generated. To achieve fine-tuning results adapted to a conversational text-to-SQL analysis, the following table 4.3 focuses on comparing previous characteristics of models developed and applied in different types of studies and benchmarks.

Table 4.3: Comparison of fine-tuning model characteristics

Model	Benchmarks tested	Learning Rate	Batch size
T5-base (Rai et al. (2023))	Spider	10^{-4}	16
STAR (Cai et al. (2022))	CoSQL	10^{-6}	80
PICARD (Qi et al. (2022))	SParC	10^{-4}	2048

Throughout the development of the best results, now focusing only in training SParC data with a T5-base solo, the first model, it is expected that several learning rates were tested by taking as the starting point values of related benchmarks like *Spider* represented

above in table 4.3, which targets similar tasks to get the best possible model value results, overcoming some resources limitations, for instance, computational running time. Thus, continuing the approach of not starting from scratch, but following table 4.3, the first batch size tried was 16 the same as shown in *Spider* benchmark, although at the end the total effective batch size ended with 32. These are composed by multiplying the gradient accumulation steps, with the value of 8, by the per device train batch size of 4. In a detailed manner, GPU small batches of 4 samples are processed while accumulating gradients during 8 steps before updating for the first time weights of the neural network architecture model. The main purpose of this strategy is to save GPU memory so that it is possible to run and avoid out-of-memory problems. Contrarily, the code takes longer to run. One ought to explore the stability that is improved by gradient accumulation steps, by reducing noise and variability when averaging gradient upgrades in multiple mini-batches, when compared to single mini-batches, consequently improving training. This technique is especially efficient with high-dimensional data at all granularity levels. Notice that all SQL components add different dimensions to data, referring to simple, medium, and advanced types of complexity in all kinds of SQL queries.

Throughout this process, the floating point is set to 32, to have higher precision, although it weights code running time. At last, warmup steps are set to 10 percent of the total training steps to grant a gradual increase in the learning rate to ensure stability when training. Again this approach was not persuaded by scratch, but rather through other studies already carried out (Margatina et al. (2021)). Those studies empirically tried different percentages of warmup steps ending up with 10 percent.

For the second model approach, different adjustments were made in the fine-tuning model allowing to understand which performance adjustments are the best in this case for the T5-base model. In this case, the floating point is set to 16 resulting in lower precision, however, it reduces memory usage and faster computation time. Thus, in this case, a fixed *warmup steps* of 500 was set for simplicity and consistency due to the model always going to the same number of *warmup steps*, meaning the same increase in learning rate during training.

Finally, complementing these two models, after fine-tuning and SQL generated, post-processing was done to improve the normalization of the queries by removing unwanted tokens, spaces, characters, and overall noise, that do not belong to traditional SQL programming language. This allows for better comparison between gold and predicted queries for further improvements on evaluation results.

4.4 Evaluation

In this research, the *SParC* is the main dataset focus of the study, since it shows cross-domain DB complexity, turning it close to real-world DB scenarios. *SParC* spotlights all dialog given by the user with utterances and their respective SQL queries, giving context to a final question and SQL query that ends a conversation. In this dataset, it is represented with 166 different domains in a traditional relational database management

systems (RDBMS) context, where each database domain has its unique schema, totaling 166 databases. The functionality of these schemas is to specify relationships, columns, and tables regarding their domains for semantic parsing. *SParC* data is a conversational text-to-SQL due to its ability to handle several interactions in a dialog context taking into account previous questions to simulate real conversations with context to bring DB information.

In the first experiment, some databases were chosen to train dialog context, where those databases have primary and foreign keys tangibly defined. The primary objective is to demonstrate the ability to generate SQL query taking into account the core component of the dialog context using the lowest possible costs in terms of time and budget. All databases will be tested after this first approach to compare results and extrapolate to all DBs benchmark analyses. From table 4.4, it is possible to check the total number of databases, tables, columns, and interactions selected from a pre-selected sample of DB for training and development sets, where the interactions are dialogues, knowing that this sample was only an intermediate step.

Table 4.4: Table, columns, and interactions selected for training and development for sample 1

Set	**Databases**	**Total Number of Tables**	**Total Number of Columns**	**Total number of Interactions**
Train	architecture, farm, small_bank_1, apartment_rentals, entertainment_awards, restaurant_1, pilot_record, cinema, climbing, railway, twitter_1, journal_committee, musical, wedding, riding_club	52	250	255
Dev	orchestra, concert_singer, tvshow, museum_visit	14	80	70

After implementing and testing the whole approach to a sample to both the T5-base model using a pre-train in *WikiSQL* and the T5-base model to train the model directly, all the data were introduced following the previous sample approaches. Ultimately, all data were not feasibly possible to be introduced simultaneously into the model due to computational power issues. To mitigate this problem a bigger sample strategy was implemented. This strategy is composed of four samples for our experiments: the sample one focus on our first sample experimentation of all, as already explained. Sample two includes the same data as sample one plus a huge amount of data maintaining the cross-domain database information. Sample three comprises the same data as sample one along with additional data not used in any other sample. Finally, sample four consisted of the core dataset in sample one and another set of additional data. Further information can be found in appendix A. Also note, that only data used in sample one is repeated in all

other samples to keep a consistent baseline and better precision on how to evaluate the impact of additional information in the same model. In summary, it is expected to have a total of eight samples, four samples of data for the T5-base model using a pre-train in *WikiSQL* and other four samples in the T5-base model to train directly the model. It is important to acknowledge that all data were used, although not all simultaneously. The four samples share the same data structure, facilitating a comparison between the two model approaches.

In the end, the evaluation metrics for *SParC* focus on Execution with Values and Exact Set Match without Values according to all benchmark data. Concerning Execution with Values, this evaluates SQL query's capacity to retrieve a value or a group of values based on the query's performance. Different scenarios may appear due to the query's construction errors resulting in incorrect or missing values.

While in the Exact Set Match without Values focuses on detecting the structure of the query by comparing SQL components like "SELECT", "FROM", and "WHERE" with development gold queries and checking if they are located in the building query. This approach goals to correctly predict structure, not necessarily the right value from the required utterance, in other words, it does not consider the values of the generated query. These analyses are done after post-processing that normalizes the generated SQL text to remove noise. Thus, tokenization comparison will be considered regarding each clause to know the correctness of each component in the query taking into consideration hardness criteria. This criteria distinguish between easy-level criteria that only contains "SELECT" and "FROM" clauses, followed by a medium-hardness level of SQL that has one or two components from these SQL components, "WHERE", "GROUP BY","ORDER BY","LIMIT","JOIN","OR","LIKE" and "HAVING". In cataloged as hard-level criteria, there are more than two and fewer than four of those components. Finally, extra-hard SQL queries level with more than three of those components.

Moreover, a Confusion Matrix regarding each component was essentially built to have true positives (TP), true negatives (TN), False Positives (FP) and False Negatives (FN) values. The way it is broken down into components provides a way to evaluate performance by classes, each clause being a class. The comparison happens between the generated SQL and the true SQL. In the end, it is possible to check the classification accuracy for each SQL clause meaning corrected identified matches, TP, also corrected identified non-matches, TN, where clauses do not exist in generated or gold SQL queries, due to the absence of an identification need from the model to generate SQL, given a specific utterance (e.g. the words in an utterance are not recognized by the model as a database name available making SQL query difficult of being generated). Thus, incorrect matches were identified as being a match, but they are no match in any way, FP, and finally matches that should have been captured, although not successful, these are the FN.

Chapter 5

Conversational Text-to-SQL Algorithm

In this chapter, two main algorithmic methods were developed to address text-to-SQL in a conversation context. The first algorithm approach emphasizes the power of schema linking, improved by using NER as a complement technique to map natural language utterances to database schema components, which is possible to visualize in the first pseudo-code structure shown in this chapter. The second algorithm, integrated into the preprocessing stage, uses dependency parsers from computation linguistics to improve the syntactic and semantics of natural language utterances in context, as shown in the second pseudo-code.

5.1 Schema linking

Consider rule-based approaches that focus on extracting linguistic most common patterns regarding, for example, grammar rules, or define entity rules to have information from utterances. Schema linking extracts entities that match the relational databases' schema. Most approaches avoid using rule-based methods and simply apply pre-trained LLM directly. As mentioned by Katsogiannis-Meimarakis and Koutrika (2023), even recent studies continued to follow only their neural network capacity to predict the text-to-SQL final query result. However, there is a large margin to improve schema linking that can be defined in the preprocessing step, leading to data structured and consequently linked to the databases. The rule-based approach for schema linking focuses mainly on defining the names of the databases, tables, and columns to prepare the data to do the mapping to reach DB.

In schema linking, one of the first steps is to understand the initial parsing through the context knowing that the potential generated SQL queries are the query candidate. In this way, it is applied to query candidate discovery by using NER to find words and phrases that are key in utterances to map possible query candidates. Those database components are column links, table links, or value links to build the correspondent database schema elements. The main objective of NER in this project is to identify the different entities in this case domains meaning several entities like people, places, and jobs that are part

of the schema mapping database. In the first pseudo-code, it is demonstrated a way to map entities that are recognized as schema elements based on manual schema mapping. Figure 5.1 provides a visual representation of the process of mapping utterances to schema elements. This approach turns out to be unique due to the integration of the *stanza* and *spaCy* libraries, promoting the advantages of both. Further analyzed by using coreference offers the possibility of complete NER and complex integration of terms being generated for answering questions on user's intent, turning this approach more complete.

Algorithm 1: Schema Linking Mapping

Input: NL text
Output: List of tuples corresponding to an entity schema elements

```
Step 1: Stanza and SpaCy with coreference
Step 2: Schema Mapping
    Schema_mapping ← {database_name, table_name, column_name}
Step 3: Map text to schema
    for token in corpus do
        if token.entity_type in schema_mapping then
            mapped_element ← SomeValue;
            if mapped_element then
                schema_elements ← AddToElementsList;
            end
        end
    end
Step 4: Return schema element identified
result ← mapped_element;
Step 5: Execute Query
output ← schema_elements;
```

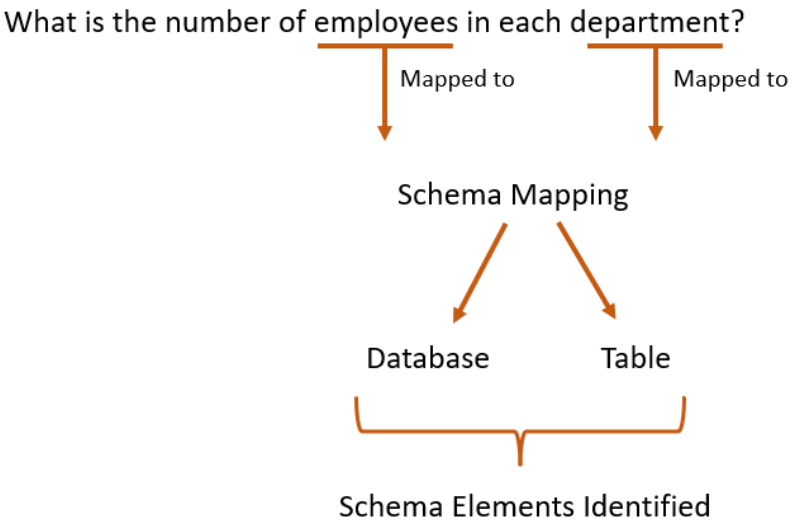

Figure 5.1: Process of mapping utterances to schema elements

All preprocessing steps will be kept to test the impacts of different model architectures implemented subsequently. This process is also known as an ablation study (Li et al. (2023a)), where it is expected to constantly remove and verify outcomes. Since the model itself already has incorporated NER, like the T5 model architecture, it is expected that with this schema linking approach on the preprocessing step, the models do not overweight running time performance considering the scalability of the data. In general, the models require less effort through schema linking developments. One of the measures used to compare performance is the F1 score to evaluate embeddings. The F1 score focuses on precision and recall (Good (1967)). Precision states a ratio of true generated SQL queries regarding the whole amount of SQL queries generated and recall targets for the ratio of correct SQL queries being generated against entirely possible true SQL queries. During this process, each query is treated as a collection of tokens and the query is compared token by token from the generated query with the gold query. The formula for F1 is presented as follows:

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5.1}$$

The score range for F1 is between 0 and 1, the closer to 1, the better the performance of the model. A score greater than 0.5 will be considered as a modest development of the schema linking code. Reinforcing that the Execution with Values and Exact Set Match without Values metrics will be also used to analyze the algorithm model results.

5.2 Dependency Parsing

Dependency parsing allows to understand the semantic and syntactic natural language of the utterances focusing on the grammatic structure to later generate the query (Schuster and Manning (2016)). This is one of the key steps to improve context interpretability that is developed interchangeably with POS tagging to achieve relationships between words in utterances. There are several types of dependency parsers, including graph-based parsers, also known as tree parsers, which have been chosen for this project (Kübler et al. (2009)). This approach turns unique due to integration with NER and incorporation with POS tagging, this second pseudo-code promotes data mapping before being inserted into the transformer models, enabling the handling of complex queries being predicted.

Concretely, all steps are generally structured in the second pseudo-code, which refers to dependency parsing defined with POS. Initially, the graph-based parsing, done with *stanza* NLP pipeline, is configured to work also with tokenization and POS tagging. Thus, a function uses as input the utterances' text to have word connections and grammatical frameworks. This is followed by defining SQL query main components, like "SELECT", "FROM" and "WHERE" clauses, among others. Inside the function's loop, every word is covered from the input through the word's dependence relationship and POS to know exactly which component role belongs to in a query. In this way, it is expected that nominal

subjects may correspond to tables' names and verbs may lead to actions and modifiers to conditions. Parsers done with *stanza* define the syntactic head of each question meaning the keyword that contains the most context, which can be a verb, a noun or any other type of grammatical property in each utterance (Qi et al. (2020)).

Algorithm 2: Text-to-SQL Dependency Parsing

Input: NL utterance
Output: SQL Query Generated

```
Step 1: Stanza pipeline parsing initialization
Step 2: Define SQL query components
    SELECT_clause ← "SELECT"
    FROM_clause ← "FROM"
    WHERE_clause ← "WHERE"
Step 3: Parsing with stanza
    for utterance in dialog do
        for token_word in utterance.tokens do
            // POS tagging and dependency parsing
            if token_word.deprel == "nsubj" or token_word.deprel == "obj"
             then
                from_clause ← f"FROM{token_word.text}"
            end
        end
    end
Step 4: Execute Query
result ← query(sql);
Step 5: Store the output in a JSON format
output ← json_output(sql);
```

Regarding *stanza* parsing, it states that an entity called an "artificial root symbol" (Qi et al. (2020)) is the starting point of a node tree. This "root symbol" anchors the tree as the starting point node, from which all other words syntactically derive as branches. Thus, the syntax root tree is essential for giving structure to parsing, and all other syntax words' roles are compared to that specific root word, which is usually a verb. In figure 5.2 and figure 5.3, there is a representation of syntax analysis examples of an utterance to detect root and headwords and a tree of graph-based dependency parsing, respectively.

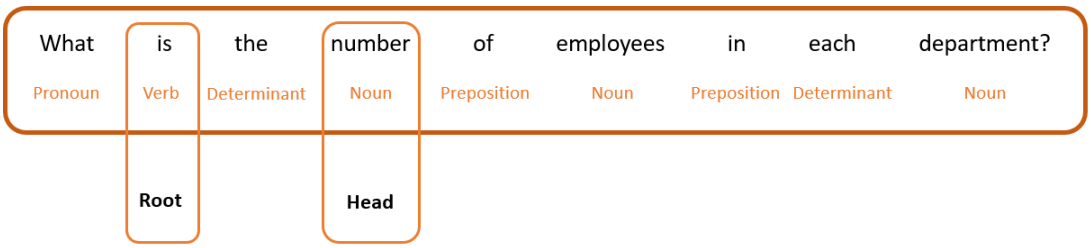

Figure 5.2: Syntax analysis example to detect root and headwords

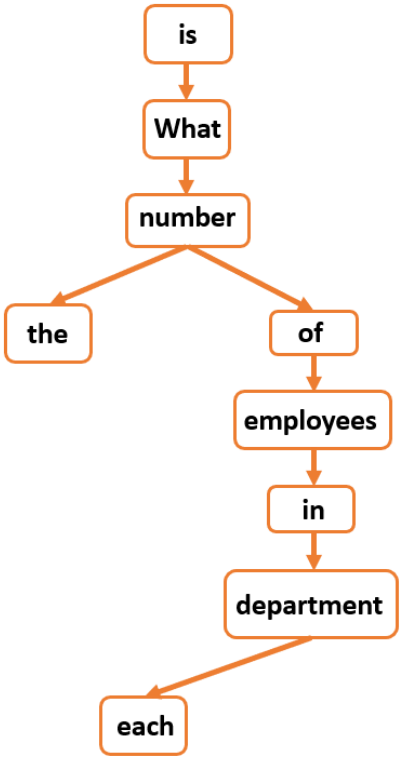

Figure 5.3: Graph-based Dependency Parsing

The granularity of parsing will be at the query level and at different database domains to expand cross-domain adaptability. To measure the loss and effectiveness of this approach, two principal metrics were selected: Execution with Values, and Exact Set Match without Values metrics, already explained in section 4.4. Also, another metric will be computed regarding component-wise accuracy, in this case, a partition of components to evaluate different clauses of SQL in a conversational text-to-SQL, namely, the "SELECT", "FROM", "WHERE", "GROUP BY", "HAVING", and "ORDER BY" clauses based on tokenization comparison. This last metric in specific can help to understand in which part of the query may need some improvements.

Chapter 6

Analysis of the Results

This chapter extensively reports the final results from the ablation study exposed in chapter 4. This section will be split into two main different analyses:

• Results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data. The pre-trained model used is *mrm8488/t5-base-finetuned-wikiSQL*.

• Results from training *SParC* data with a pre-train in T5-base model with fine-tuning. The pre-trained model used is *t5-base*.

At the end, a comparison between those approaches will be presented regarding the execution with values and the exact set match without values. Both metrics were the chosen ones in the official leaderboard [1], as such, they are the best evaluation metrics to compare results.

6.1 Pre-trained *WikiSQL* with fine-tuning in *SParC* data

6.1.1 Execution with values

In the first experiment, only a sample of DBs was selected as mentioned in table 4.4 to save financial and computational resources. Maintaining the same pre-processing strategy and methodology for all samples' data, the results can be compared according to different degrees of success, as the model shows some difficulties in generalizing well to samples 1, 3, and 4. These samples present a loss of accuracy when tested on some unseen data when predicting true positives. Inversely, results improve on sample 2, as shown in table 6.1, for execution with values metric.

Table 6.1: Samples with Train and Dev data SQL matches

Samples	Train	Dev
Sample 1	0.5%	0.0%
Sample 2	5.1%	5.1%
Sample 3	31.5%	6.7%
Sample 4	33.9%	5.1%

Upon execution of conversational text-to-SQL with a pre-trained *WikiSQL* with fine-tuning in *SParC* data, in the execution with values for all samples. In general, all samples shows low results due to identified errors in most SQL clauses, especially the "HAVING" and "GROUP BY" clauses. These errors impossibilities the SQL engine from predicting well the conditions specified to the model through the algorithms, thereby wrongly interpreting by the model. To address this issue, during successive attempts, post-processing gains great focus to improve the normalization of the generated SQL. These processes focus on turning SQL into the most rigorous data possible, after analyzing the results, however without significant success for the model to reach good final value results of true positive sample predictions.

Comparing fine-tuning settings, for the sample 1, it was defined 40 epochs due to a small amount of data, a reduced number of epochs was required for the model to learn in training showing a 0.085200 of loss result. When the same procedure was extrapolated to "all data" samples (sample 2,3 and 4), more features were taken into consideration to avoid out-of-memory errors, which began to be more likely to appear with the increase in data requiring more CUDA memory resources. In the samples' experimentation fine-tuning settings, no gradient accumulation steps were taken into consideration, it was only applied as a solution with t5-base solo model experiments to save GPU memory.

By comparing the different settings trained in a sample 1 versus training all the data samples 2,3 and 4 using the same base approach, it is expected to understand the results respectively adapted to these different definitions. However, as the results are shown in table 6.1, for the sample 1 and for all the remaining data samples, all results presented are not satisfactory.

6.1.2 Exact set match without values

The exact set match, without values for question match, compares the tokenization of the SQL query generated with the gold query regarding pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data. Since the final results were not the most satisfactory, a partition of SQL query components was required to understand where the model predicts well and where it struggles to make those predictions. The mainstream evaluation regards question match, facilitating an accuracy comparison for each SQL component for translating NL queries into SQL queries.

Respectfully to loss results, as presented in appendix C, using all sample data (sample

[1] https://yale-lily.github.io/sparc

1, 2, 3 and 4), visible in table 6.2, for training, the loss best value result is 0.017500, from sample 4, and for the validation set is 0.022857, from sample 3, which means that, in general, the model is predicting well to unseen data. This scenario demonstrates no overfitting to the train data, in other words, the model does not experience undue interference with the amount of noise present in the training data driving it to good performance results for unknown data. Also, in table 6.2, it is possible to check the model outcomes that only use the sample data with their loss results. Comparing the same model with different amounts of data for loss results, in this situation, more data allowed the model to decrease loss, which means the best accuracy prediction results from the model when data is added to the model.

Table 6.2: Best Training and Validation Loss for Each Sample

Sample	Best Training Loss	Best Validation Loss
Sample One	0.0849	0.078165
Sample Two	0.0192	0.024336
Sample Three	0.0187	0.022857
Sample Four	0.0175	0.023978

Since the tokenization comparison analyzes the entire queries for being generated all at once, the results achieved were not satisfactory, a comparison with partition SQL components, namely ”SELECT”, ”FROM”, ”WHERE”, ”GROUP BY”, ”HAVING”, and ”ORDER BY” clauses gain a particular focus on how to evaluate results.

When we compare the results from the sample 1 data evaluated and all data samples 2,3 and 4, it is clear that the components with the best results are the ”SELECT” and ”FROM” clauses for all kinds of sampling data used. In table 6.3, results are shown using the same base model even fine-tuned with slight differences for different SQL components, enabling an easier visual way of where the model makes better predictions. In absolute terms, more data, in this case, means more correct predictions were made, knowing that the model makes better predictions for ”SELECT” and ”FROM” clauses, whose components are part of the easy-level category of hardness criteria. It can be seen from table 6.4 that the majority of SQL queries have low or medium levels of complexity, then, it is expected greater probability of obtaining better results in the components of ”SELECT” and ”FROM” clauses that naturally make up these levels of hardness criteria.

Table 6.3: Total Number of Matches for Each Clause by Sample

Clause	Sample 1	Sample 2	Sample 3	Sample 4	Total
SELECT	71	244	384	450	1149
FROM	60	1045	849	1208	3162
WHERE	6	60	28	63	157
GROUP BY	0	0	0	0	0
HAVING	0	0	0	0	0
ORDER BY	0	19	15	17	51

Table 6.4: Total Number of Hardness Criteria by Sample

Category	Sample 1	Sample 2	Sample 3	Sample 4	Total
Easy	33	908	1051	118	2110
Medium	53	2307	2241	290	4891
Hard	10	514	643	90	1257
Extra Hard	10	461	535	85	1091

In section 4.4 details the confusion matrix evaluation possible to find in appendix B. Considering the best scenario, regarding true positives that are the well-predicted SQL query component, where table 6.3 shows these values already described.

In the opposite way, considering the worst scenario in these confusion matrices, where there are a lot of false positives meaning wrong predictions being made indicating inaccuracies in the training process. These false predictions can lead to users' wrong decisions when using this information, in this specific case only components of SQL queries are being considered, indicating, in the end, no danger or any other concern since no total queries are fully being considered in this situation, so no final prediction, only partially predictions. Finally, no false negatives were detected in any of the samples, solely allowing capturing outputs from the model when queries were correctly predicted.

As a final point, the F1 score for the six SQL components mentioned before, along with four samples, is composed of precision and recall. Generally, precision shows high results for some clauses, in detail, with regard to "SELECT", and "FROM" clauses, frequently indicating a correct match prediction. However, recall reaches very low values for "WHERE", "GROUP BY", "HAVING", and "ORDER BY" clauses, generally leading to poor F1 score results, traducing the instability between precision and recall.

6.1.3 Time execution

There are four key components in terms of time execution, designating the megabytes size of the model, followed by the dimensional vectors created through the model, plus SQL query runtime, on average, and seconds for each query being generated, and the total runtime for generating all SQL queries.

Concerning the model, that is equal in all four samples regarding pre-training in the T5-base model with *WikiSQL* data, later fine-tuned with *SParC* data, those samples experiment on the model that totals 850.31 megabytes in size with 768-dimensional vectors.

Also, on average, the sample's runtime per SQL query being generated is different from each sample due to different amounts of data introduced in the model and the implicit randomness of each model when generating their final output values. Sample 1 contains the base data that is also included in other samples 2,3, and 4, having an average, of 5.2094 seconds to generate a SQL query while sampling 2 has an average time of 1.8144 seconds, sample 3, around 1.2798 seconds, and finally sample 4 with a mean of 1.8602 seconds to produce an SQL query output. In conclusion, on average, sample 1 takes more

seconds to generate an SQL query, which is counterintuitive, because with less data it would be expected to generate fast SQL queries, although, this runtime is just a mean, and it is possible that in sample 1 it is more complex SQL queries being generated taking more time than the average. In real-life terms, this time would be the waiting time for a user after asking a question to the database.

In terms of total runtime for generating all SQL queries, the model took 26.0532 seconds for sample 1, 9.0768 seconds for sample 2, 5720.8586 seconds for sample 3, and 9.3065 seconds for sample 4. Sample 3 shows a huge amount of data due to the highest amount of time for all queries being developed.

6.2 Train all *SParC* data

6.2.1 Execution with values

Considering true positive results from training *SParC* data with a pre-train in the T5-base model with fine-tuning, execution results with values can be seen in table 6.5 with train and development datasets knowing that an early stopping of ten epochs was implemented to guarantee an automatic stopping. In case the model results do not improve for those number of epochs, consequently, this method prevents from overfitting and increases the effectiveness of training the T5-model. The values presented in table 6.5 show underfitting specially for 2, 3, and 4 samples, which might happen due to the model did not learn enough from the data to give good results, in other words, this suggests that the T5-model is too simple for a conversational text-to-SQL or the data is too noisy or the model did not train long enough. In detail, all samples were trained with 60 epochs taking about more than 2 hours and a half each, where only sample 1 stopped two epochs before otherwise, the other samples 2, 3, and 4 continued by proceeding until reaching 60 epochs, meaning more time and epochs required. However, at 60 epochs, the loss results were already low enough to believe in further good results. Sample 1 commits behavior of overfitting due to a high decrease from the training to development sets, caused by a small volume of data driven through limited memorization of data patterns, guiding to poor generalization.

Table 6.5: Samples with Train and Dev data SQL matches

Samples	Train	Dev
Sample 1	11.5%	0.0%
Sample 2	10.8%	2.1%
Sample 3	4.4%	2.2%
Sample 4	12.0%	2.2%

In this model, a gradient accumulation step was considered for memory management regarding CUDA usage savings and preventing out-of-memory failures. A larger batch size, in this scenario, simulates a hardware artificial expansion to the system handle a

huge batch size and accumulates over 8 steps. Meanwhile all of these steps, the weights are just updated at the end, when the batch size multiplies by the accumulation steps under these circumstances reaching 32 final batch size.

During training, a warm-up was set initially to promote the constant increase of learning until rise the final peak value disabling abrupt variations. After hitting the top, the decay of the learning rate to levels that allow parameters to have proper finetuning adaptation for conversational text-to-SQL problems. These characteristics were added to the model to guarantee model stability. Therefore samples are being used as a strategy to overcome financial resource problems, then multi-GPU training turns out to be unnecessary.

6.2.2 Exact set match without values

In this section, the exact set match without values results measure is under analysis that considers the comparison between the tokens of the utterances and the gold query in training with *SParC* data with a pre-train in the T5-base model with fine-tuning. In relation to loss results, as illustrated in appendix C, all sample results are shown in table 6.6, where only the loss best model training values are kept to compare each sample against the validation loss best result, in this case, sample 4 presents the lowest final value for validation loss of the model results, which means a better fit of the model performance in terms of generalization. In each sample, the gap between the train and validation loss is short release the idea of no overfitting meaning a good generalization of the model in new data. Notice that all samples have different sizes, domains, and specialized content of data from distinct databases.

Table 6.6: Best Training and Validation Loss for Each Sample

Sample	Best Training Loss	Best Validation Loss
Sample 1	0.0566	0.048361
Sample 2	0.0582	0.048685
Sample 3	0.0574	0.049588
Sample 4	0.0568	0.046416

Since the final results of the exact match are not promising values, another approach was taken into consideration meaning the number of matches through different components of the SQL query, ”SELECT”, ”FROM”, ”WHERE”, ”GROUP BY”, ”HAVING”, and ”ORDER BY”. In table 6.7, the ”SELECT” and ”FROM” clauses have the highest frequency of well-predicted matches for utterances, knowing that the ”FROM” clause achieved the highest of all. Contrarily, in more complex clauses that include ”GROUP BY”, ”HAVING”, and ”ORDER BY”, the model does not predict correctly, demonstrating a lack of efficiency on advanced queries in all samples. Table 6.8 shows most of the SQL queries are easy and medium levels achieving ”Easy” or ”Medium” categories

in terms of query composition. The complex queries' structures were enduring the most adversity from model predictions.

Table 6.7: Total Number of Matches for Each Clause by Sample

Clause	Sample 1	Sample 2	Sample 3	Sample 4	Total
SELECT	76	313	219	399	1007
FROM	123	372	197	563	1255
WHERE	6	21	12	50	89
GROUP BY	0	0	0	0	0
HAVING	0	0	0	0	0
ORDER BY	0	0	0	0	0

Table 6.8: Total Number of Hardness Criteria by Sample

Category	Sample 1	Sample 2	Sample 3	Sample 4	Total
Easy	33	908	1051	118	2110
Medium	53	2307	2241	290	4891
Hard	10	514	643	90	1257
Extra Hard	10	461	535	85	1091

In appendix B, there are several confusion matrices detailing not only the information possible to find in table 6.7 regarding true positives, but also false positives, true negatives, and false negatives, as explained in more detail in section 4.4. It is important to note that in conversational text-to-SQL, false positives are the greatest challenge as they can lead users to obtain wrong information in the end. If the model focuses only on some components it is easy to not get lost in the context of the utterances, because the target is not the all questions but rather only the components being predicted at once, in this way, it can reduce understanding ambiguities. This high-level granularity brings concrete focus and detail to small pieces of the SQL queries pinpointing specific errors that were essential to adapt in the post-processing. Sample 4, presents 1012 true positives for all components, which is not a high number of correct predictions against 1674 false negatives, implying the model misses such a high number of actual positive predictions. The model did not produced incorrectly predictions to any negative instance as positive, meaning there were no false positives results predicted by the model. All in all, this model shows lower precision.

The F1 score for "SELECT" and "FROM" clauses present high precision, however very low recall results. In simpler terms, higher precision means a low rate of false positives, accordingly, the model accurately predicts the clause. While low recall, advice for several model inaccuracies, when detecting all possible predictions. On the opposite side, "GROUP BY", "HAVING", and "ORDER BY" SQL components show zero precision and recall, which means that clauses are not well structured to bring correct identification results. This metric reinforces the results of the previous metrics.

6.2.3 Time execution

The model's megabyte size, the dimensional vectors created through the model, thus the average SQL query runtime in seconds for each question being generated, and the total runtime for generating all SQL queries are the four main measures that traduce how quickly an output can be executed to generate a SQL query.

Regarding the model, all four samples have the same training *SParC* data with a pre-train in the T5-base model with fine-tuning, those experiments used samples of data to place inside the model, which totaled 850.31 megabytes in size with 768-dimensional vectors. Furthermore, the different samples have different results in runtime to generate each SQL query, on average, the model required 0.2161 seconds in sample 1, while sample 2 estimates 0.1472 seconds, sample 3 needs about 0.2157 seconds, and finally sample 4 took 0.2080 seconds. Overall, sample 2 has reached less time compared to other samples. However, the difference between all samples is no more than around 0.07 seconds, also notice these values change over the randomness of neural network models like T5.

In conclusion, the model took 1.0878 seconds to generate all SQL queries for sample 1, 0.7459 seconds for sample 2, 1.0860 seconds for sample 3, and 1.0469 seconds for sample 4. Again sample 1 took the most runtime with less data introduced in the model compared to other samples' data experiments, due to model's effort to adjust their system's design and optimize processing steps to handle small data. This impact may not be noticeable with higher data volumes.

6.3 Comparing methodologies and Improve outcomes

These methodologies were run in NVIDIA RTX 6000 Ada Generation machine with one GPU available meaning no parallelization was required, knowing that each sample required a different GPU RAM of 0.5 GB for sample 1, the smallest sample of all, 23.3 GB for sample 2, 23.3 GB for sample 3, and 23.2 GB and sample 4, regarding the model that has a pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data. In addition, the total number of CPUs per sample is 16.0 CPUs for sample 1, 32.0 CPUs for sample 2, 32.0 CPUs for sample 3, and 24.0 CPUs for sample 4. While the model trains *SParC* data with a pre-train in the T5-base model with fine-tuning using the same NVIDIA RTX 6000 Ada Generation machine required 12.8 GB of GPU RAM for sample 1, 24.4 GB for sample 2, 24.2 GB for sample 3, and 24.2 GB for sample 4. Lastly, concerning the total number of CPUs, sample 1 needed 12.0 CPUs, 24.0 CPUs for sample 2, 24.0 CPUs for sample 3, and 24.0 CPUs for sample 4.

The crucial differences between methodologies are focusing on how larger pre-trained and fine-tuning steps are used to possibly benefit from transfer learning along *WikiSQL*-related tasks versus another methodology training directly on target task, *SParC*, needing more data or iteration epochs to attain similar performance. On closer examination, from previous results described before and focusing on table 6.2 and table 6.6 results, a methodology that proved to achieve better results is the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data when compared to sample results

from training *SParC* data with a pre-train in T5-base model with fine-tuning. This means that the pre-training in the T5-base model with *WikiSQL* is being more refined and adapted to this problem. Finally, this scenario happens in an unexpected behavior, because a model producing better results being trained with different data than the target data, primarily would result in poor results compared to a model being immediately trained with the target data, at the end of this study, results prove the opposite situation. Although, both results were not successful, due to several problems. Of the various problems, NER could be considered the main issue.

Enhancing NER for huge amounts of data. Initially, as presented in this study project was conducted on a small sample (sample 1) that when generalized to multiple databases with identical names, entity duplication concerns must be improved in order to scale well to all data. While *Namespace Prefixing* was tested to mitigate these problems, models such as T5-base are better off with simpler entity references. Moreover, it can be difficult to handle aliases in queries, particularly those that are alphanumeric, which makes entity identification more difficult.

As a culmination of all these observations, selecting only one model due to their performance, the preferable model option would be pre-training in the T5-base model with *WikiSQL* data later fine-tuned with *SParC* data model. This model is already prepared to generate SQL queries because the pretraining was already optimized for this task turning it into a model prone to show more accurate results.

Chapter 7

Conclusions

This project's main objective was the development of conversational text-to-SQL, which, as it is possible to see through the values of the results of the models, there is still a lot of room to continue evolving to improve current approaches. Despite the progress made with previously established methods, challenges remain in developing conversational text-to-SQL algorithms.

The two algorithms developed gain the main focus scenario for the improvement of results since these algorithms bring a lot of information from syntactic grammar analysis, with tree dependency parsing to obtain information about the grammatical structure of the sentence and their relationship between words. In addition, lexical analysis from NER and POS, which improves the context and understanding of utterances. Also, NER promotes the categorization of entities that improves semantic analysis for SQL generation.

However, mapping all entities for all different SQL databases, tables, and columns from distinct domains, turns it into a challenge for cross-domain adaptability and execution time increases significantly. An ablation study was performed by taking the preprocessing steps previously mentioned with a comparison between T5-base model trained with *SParC* data and the inference from *WikiSQL* later finetuned for *SParC* data. The main constraint of these models was the high computational power demanded, which could benefit from an even larger T5 model, although difficult to pursue with the real resources available, meaning a more efficient server is not purchasable for a single individual. This research project would benefit from a future partnership with a company capable of paying for higher computational power.

According to section 6.3, it is feasible to answer the research question mentioned in the introduction, a recap is provided:

• Research question - "How can a pre-train model be used to help the development of semantic parsing and taxonomy improvements in a large-scale cross-domain context-dependent dataset?". This research question addresses two distinct themes, taxonomy, and semantic parsing, where taxonomy is widely known as schema linking. Both were treated in two distinct algorithms as referred to before. The results regarding the complete execution with values stated as unsuccessful results due to the inability of the queries to be fully correct in terms of SQL code preventing any value source from being achieved. In the metric regarding exact set match without values, when split into SQL query components, the "SELECT" component achieves the best results in most of the samples' experiments. However, in the overall results of metrics, when comparing the two models,

the model pre-trained on the T5-base architecture using the *WikiSQL* data, and later fine-tuned with *SParC* data, achieved the best results when compared to a model trained on the *SParC* data with pre-training on T5-base and fine-tuning. But, none of them demonstrated satisfactory results.

The final results of the algorithms were not satisfactory, even with low results in terms of loss, demonstrating the space to improve limitations regarding conversational text-to-SQL. In the final analysis, the two algorithm approaches helped enhance the context and the understanding of the questions by applying this new approach and improving taxonomy. However, some challenges appeared, revealing future limitations to be improved. In this regard, the next section presents the main limitations found in this dissertation project.

7.1 Future work and algorithms limitations

The main issue suppresses targeting entities correctly with their respective names of the databases followed by their respective tables and written columns, and lack of computational resources. Focusing on those restrictions, these are limitations found for future work:

• Hybrid approach - In this project development, several NLP techniques were implemented starting in the preprocessing step with NER, dependency parsing, and POS, followed by finetuning T5-base models, resulting in a hybrid approach. To mitigate some of the flaws found throughout this process, the following limitations are explained in detail:

◇ Improve NER - The first experiments were done with a sample, where NER had fewer entities to create the path to reach the DB data. With the extension to all data, care was taken not to repeat entities, as with more data there is a greater possibility of table names being the same, but referencing a different database and consecutively different data. Typically, the table name influences or gives rise to the entity name. An alternative found to improve that issue was creating NER with *Namespace Prefixing*, however, for the model it is better the more straightforward the entity is. For the T5-base model, just one word referencing the entity would be ideal, because this facilitates the correspondence between the question word and the entity word, knowing that users will not write questions with *Namespace Prefixing*. Also, when queries evolve aliases the behavior of the code generated presents the worst results, it is more difficult for the model to understand the name given in the aliases. To what extent do these aliases refer to, given that most are combinations of letters and numbers like ”T2”. It is difficult for the model to determine which database or column aliases refer to which entities. A possible future solution is a hierarchical NER approach, where general categories are defined as the first entities later honed by more specific entities corresponding to detailed categories taking into account the cross-domain adaptability for different DB themes.

◇ Improve tree dependency parsing - The most efficient APIs proving high levels of accuracy results are mandatory to purchase, which can be an obstacle for individuals to acquire. For example, Google's Cloud Natural Language API focuses on several NLP tasks like syntax analysis, which contains dependency parsing. Despite the existence of free tools, they do not have the same effectiveness and consequently have weaker final results.

◇ Improve context question answer - There are some NLP tasks that were not performed to preserve context, for example, stop words. In this specific case of a conversational text-to-SQL, using transformers may have the opposite effect, as this specific task requires keeping context interactions. The context may be compromised by adding those removals, the same behavior with lemmatization. Under these circumstances, the solution goes through the possibility of using better models like T5-11b that entail substantial resources, meaning a GPU with more RAM. This project thrives by leveraging advanced computational models.

• CRUD (Create, read, update, and delete) actions - A conversational text-to-SQL requires read-only permissions widely used by business analysts daily to create SQL queries. Although SQL code language is more extensive in all possible tasks regarding writing permissions, such as create, update, and delete actions, those tasks are mainly done by information management officers. Create command may belong to the creation of DB or tables or the insert of more lines of values inside tables. The insert function may have quality test abilities, among many other operations.

• Relation-aware attention mechanism - Integrating the relation-aware attention mechanism inside the model architecture is essential to capture the relationship in the text of the utterances by adding layers or heads in model design. The training data would need to be learned from the integration of relation labels or manual annotations regards column names and other elements in the dataset.

These issues may be solved in the future with the improvement of current models requiring more computational power and future financial investments. It is speculated that these new, more sophisticated models are developed by large companies, following the pattern we have seen to this day, such as *OpenAI* and *Google*, making it difficult for an individual to purchase huge computational power.

7.2 Limitations of the *SParC* benchmark

In all interactions done between question and answer, there is no general feedback from the user in demonstrating whether the previous question was answered correctly or not, which makes it difficult to apply a reward in case of a well-succeeded query generated through the context given. In this way, methodologies like the application of reinforcement learning with rewards would be difficult in benchmark data like *SParC*.

The principal evaluation metrics in the official leaderboard are execution with values and exact set match without values, which is a binary evaluation metric focusing on all final query output results generated, classifying them as either correct or incorrect predictions. However, the evaluation should instead account as a way of doing partial correctness evaluation. Despite not being part of the leaderboard, in this project, partial evaluation by component was carried out to know exactly where the model fails to predict. In table 7.1, there are detailed mistakes found in the dataset regarding gold query, the basis for the model being trained that is highly affected by the quality of the data.

Table 7.1: *SParC* benchmark mistakes in gold queries

Utterance	Gold query	Comments
Which one has the highest number of flights?	`SELECT T1.AirportCode FROM AIRPORTS AS T1 JOIN FLIGHTS AS T2 ON T1.AirportCode = T2.DestAirport OR T1.AirportCode = T2.SourceAirport GROUP BY T1.AirportCode ORDER BY count(*) DESC LIMIT 1`	The SQL query could benefit from parenthesis on the ”ON” clause to avoid errors when generating more complex queries.
Give the airport code and airport name corresonding to the city Anthony.	`SELECT AirportCode, AirportName FROM AIRPORTS WHERE city = "Anthony"`	Wrong spell of the word ”corresonding” in the utterance.

In conclusion, the real world always presents complex scenarios difficult to show in data to train due to complex queries and dataset size limitations. Hence, by not capturing the full diversity of all scenarios, it is constrained to generalize well and generate a robust generalization, especially if the user's language for unseen data integrates slang or regional dialect in utterances.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Arevalillo-Herráez, M., Arnau-González, P., and Ramzan, N. (2022). On adapting the diet architecture and the rasa conversational toolkit for the sentiment analysis task. *IEEE Access*, 10:107477–107487.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, 13.

Benmalek, R. Y., Khabsa, M., Desu, S., Cardie, C., and Banko, M. (2019). Keeping notes: Conditional natural language generation with a scratchpad mechanism. *arXiv preprint arXiv:1906.05275*.

Bhaskar, A., Tomar, T., Sathe, A., and Sarawagi, S. (2023). Benchmarking and improving text-to-sql generation under ambiguity. *arXiv preprint arXiv:2310.13659*.

Brants, T. (2000). Tnt-a statistical part-of-speech tagger. *arXiv preprint cs/0003055*.

Bromuri, S., Zufferey, D., Hennebert, J., and Schumacher, M. (2014). Multi-label classification of chronically ill patients with bag of words and supervised dimensionality reduction algorithms. *Journal of biomedical informatics*, 51:165–175.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Byrd, R. J. and Ravin, Y. (1999). *Identifying and extracting relations in text*. na.

Cai, Z., Li, X., Hui, B., Yang, M., Li, B., Li, B., Cao, Z., Li, W., Huang, F., Si, L., et al. (2022). Star: Sql guided pre-training for context-dependent text-to-sql parsing. *arXiv preprint arXiv:2210.11888*.

Castelle, M. (2018). The linguistic ideologies of deep abusive language classification. In *Proceedings of the 2nd workshop on abusive language online (ALW2)*, pages 160–170.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chieu, H. L. and Ng, H. T. (2002). Named entity recognition: a maximum entropy approach using global information. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Choi, D., Shin, M. C., Kim, E., and Shin, D. R. (2021). Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics*, 47(2):309–332.

Chowdhary, K. (2020). *Fundamentals of artificial intelligence*. Springer.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Colmerauer, A. and Roussel, P. (1996). The birth of prolog. In *History of programming languages—II*, pages 331–367.

Dai, B., Li, J., and Xu, R. (2020). Multiple positional self-attention network for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7610–7617.

Dalal, A., Nagaraj, K., Swant, U., Shelke, S., and Bhattacharyya, P. (2007). Building feature rich pos tagger for morphologically rich languages: Experience in hindi. *ICON*.

Eisenschlos, J. M., Gor, M., Müller, T., and Cohen, W. W. (2021). Mate: multi-view attention for table transformer efficiency. *arXiv preprint arXiv:2109.04312*.

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

Fu, H., Liu, C., Wu, B., Li, F., Tan, J., and Sun, J. (2023). Catsql: Towards real world natural language to sql applications. *Proceedings of the VLDB Endowment*, 16(6):1534–1547.

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., and Zhou, J. (2023). Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Gkini, O., Belmpas, T., Koutrika, G., and Ioannidis, Y. (2021). An in-depth benchmarking of text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*, pages 632–644.

Good, I. (1967). The decision-theory approach to the evaluation of information-retrieval systems. *Information Storage and Retrieval*, 3(2):31–34.

Hazoom, M., Malik, V., and Bogin, B. (2021). Text-to-sql in the wild: a naturally-occurring dataset based on stack exchange data. *arXiv preprint arXiv:2106.05006*.

Hernandez, N., Lundström, J., Favela, J., McChesney, I., and Arnrich, B. (2020). Literature review on transfer learning for human activity recognition using mobile and wearable devices with environmental technology. *SN Computer Science*, 1:1–16.

Herzig, J., Nowak, P. K., Müller, T., Piccinno, F., and Eisenschlos, J. M. (2020). Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

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

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th annual meeting of the association for computational linguistics (Volume 1: Long papers)*, pages 873–882.

Hutchins, J. (1994). Machine translation: History and general principles. *The encyclopedia of languages and linguistics*, 5:2322–2332.

Hutchins, W. J. (2004). The georgetown-ibm experiment demonstrated in january 1954. In *Conference of the Association for Machine Translation in the Americas*, pages 102–114. Springer.

Jatav, V., Teja, R., Bharadwaj, S., and Srinivasan, V. (2017). Improving part-of-speech tagging for nlp pipelines. *arXiv preprint arXiv:1708.00241*.

Jha, D., Ward, L., Yang, Z., Wolverton, C., Foster, I., Liao, W.-k., Choudhary, A., and Agrawal, A. (2019). Irnet: A general purpose deep residual regression framework for materials discovery. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2385–2393.

Kacprzyk, J. and Zadrozny, S. (2010). Computing with words is an implementable paradigm: Fuzzy queries, linguistic data summaries, and natural-language generation. *IEEE Transactions on Fuzzy Systems*, 18(3):461–472.

Kalyan, K. S. (2023). A survey of gpt-3 family large language models including chatgpt and gpt-4. *Natural Language Processing Journal*, page 100048.

Kang, Y., Cai, Z., Tan, C.-W., Huang, Q., and Liu, H. (2020). Natural language processing (nlp) in management research: A literature review. *Journal of Management Analytics*, 7(2):139–172.

Katsogiannis-Meimarakis, G. and Koutrika, G. (2021). A deep dive into deep learning approaches for text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2846–2851.

Katsogiannis-Meimarakis, G. and Koutrika, G. (2023). A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, pages 1–32.

Katsogiannis-Meimarakis, G. G. (2023). Data democratisation with deep learning: Structured query translation from and to natural language.

Khurana, D., Koli, A., Khatter, K., and Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744.

Kim, H. E., Cosa-Linan, A., Santhanam, N., Jannesari, M., Maros, M. E., and Ganslandt, T. (2022). Transfer learning for medical image classification: a literature review. *BMC medical imaging*, 22(1):69.

Kübler, S., McDonald, R., and Nivre, J. (2009). Dependency parsing. In *Dependency parsing*, pages 11–20. Springer.

Kumar, A., Nagarkar, P., Nalhe, P., and Vijayakumar, S. (2022). Deep learning driven natural languages text to sql query conversion: A survey. *arXiv preprint arXiv:2208.04415*.

Lee, J., Seo, S., and Choi, Y. S. (2019). Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing. *Symmetry*, 11(6):785.

Li, H., Zhang, J., Li, C., and Chen, H. (2023a). Decoupling the skeleton parsing and schema linking for text-to-sql. *arXiv preprint arXiv:2302.05965*.

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. (2001). Natural language processing.

Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., et al. (2023). Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, page 100017.

Lu, Y., Rai, H., Chang, J., Knyazev, B., Yu, G., Shekhar, S., Taylor, G. W., and Volkovs, M. (2021). Context-aware scene graph generation with seq2seq transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15931–15941.

Margatina, K., Barrault, L., and Aletras, N. (2021). On the importance of effectively adapting pretrained language models for active learning. *arXiv preprint arXiv:2104.08320*.

Martinez, A. R. (2010). Natural language processing. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(3):352–357.

Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.

Nassar, A., Livathinos, N., Lysak, M., and Staar, P. (2022). Tableformer: Table structure understanding with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4614–4623.

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Barnes, N., and Mian, A. (2023). A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

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

Parthasarathi, S. H. K., Zeng, L., and Hakkani-Tür, D. (2023). Conversational text-to-sql: An odyssey into state-of-the-art and challenges ahead. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In Walker, M., Ji, H., and Stent, A., editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, New York. ACM.

Qaiser, S. and Ali, R. (2018). Text mining: use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181(1):25–29.

Qi, J., Tang, J., He, Z., Wan, X., Cheng, Y., 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*.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

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*.

Raganato, A. and Tiedemann, J. (2018). An analysis of encoder representations in transformer-based machine translation. In Linzen, T., Chrupała, G., and Alishahi, A., editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.

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 Belz, A. (2009). An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Reshamwala, A., Mishra, D., and Pawar, P. (2013). Review on natural language processing. *IRACST Engineering Science and Technology: An International Journal (ESTIJ)*, 3(1):113–116.

Rubin, O. and Berant, J. (2020). Smbop: Semi-autoregressive bottom-up semantic parsing. *arXiv preprint arXiv:2010.12412*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., and Flickinger, D. (2002). Multiword expressions: A pain in the neck for nlp. In *Computational Linguistics and Intelligent Text Processing: Third International Conference, CICLing 2002 Mexico City, Mexico, February 17–23, 2002 Proceedings 3*, pages 1–15. Springer.

Sai, A. B., Mohankumar, A. K., and Khapra, M. M. (2022). A survey of evaluation metrics used for nlg systems. *ACM Computing Surveys (CSUR)*, 55(2):1–39.

Sathick, K. J. and Jaya, A. (2015). Natural language to sql generation for semantic knowledge extraction in social web sources. *Indian Journal of Science and Technology*, 8(1):1.

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

Schuster, S. and Manning, C. D. (2016). Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Skrodelis, H. K., Romanovs, A., Zenina, N., and Gorskis, H. (2023). The latest in natural language generation: Trends, tools and applications in industry. In *2023 IEEE 10th Jubilee Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, pages 1–5. IEEE.

Song, H., Rajan, D., Thiagarajan, J., and Spanias, A. (2018). Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Sui, G., Li, Z., Li, Z., Yang, S., Ruan, J., Mao, H., and Zhao, R. (2023). Reboost large language model-based text-to-sql, text-to-python, and text-to-function–with real applications in traffic domain. *arXiv preprint arXiv:2310.18752*.

Sukthanker, R., Poria, S., Cambria, E., and Thirunavukarasu, R. (2020). Anaphora and coreference resolution: A review. *Information Fusion*, 59:139–162.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. (2019). Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450.

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

Tsai, C.-F. (2012). Bag-of-words representation in image annotation: A review. *International Scholarly Research Notices*, 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vuckovic, J., Baratin, A., and Combes, R. T. d. (2020). A mathematical theory of attention. *arXiv preprint arXiv:2007.02876*.

Wang, B., Gao, Y., Li, Z., and Lou, J.-G. (2023). Know what i don't know: Handling ambiguous and unknown questions for text-to-sql. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5701–5714.

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, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Williams, A., Nangia, N., and Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Xiao, D., Chai, L., Zhang, Q.-W., Yan, Z., Li, Z., and Cao, Y. (2022). Cqr-sql: Conversational question reformulation enhanced context-dependent text-to-sql parsers. *arXiv preprint arXiv:2205.07686*.

Xu, 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*.

Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., and Radev, D. (2018a). Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task. *arXiv preprint arXiv:1810.05237*.

Yu, T., Zhang, R., Er, H. Y., Li, S., Xue, E., Pang, B., Lin, X. V., Tan, Y. C., Shi, T., Li, Z., et al. (2019a). Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv preprint arXiv:1909.05378*.

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*.

Yu, T., Zhang, R., Yasunaga, M., Tan, Y. C., Lin, X. V., Li, S., Er, H., Li, I., Pang, B., Chen, T., et al. (2019b). Sparc: Cross-domain semantic parsing in context. *arXiv preprint arXiv:1906.02285*.

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, R., Yu, T., and Klein, D. (2020a). Semantic evaluation for text-to-sql with distilled test suites. *arXiv preprint arXiv:2010.02840*.

Zhong, V., Lewis, M., Wang, S. I., and Zettlemoyer, L. (2020b). Grounded adaptation for zero-shot executable semantic parsing. *arXiv preprint arXiv:2009.07396*.

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

Composition of samples 1, 2, 3, and 4

Table A.1: Train sample databases for Sample 1 and Sample 2

Set	Sample 1	Sample 2
Train	architecture, farm, small_bank_1, apartment_rentals, entertainment_awards, restaurant_1, pilot_record, cinema, climbing, railway, twitter_1, journal_committee, musical, wedding, riding_club	architecture, farm, small_bank_1, apartment_rentals, entertainment_awards, restaurant_1, pilot_record, cinema, climbing, railway, twitter_1, journal_committee, musical, wedding, riding_club, hospital_1, game_1, college_2, college_1, inn_1, customers_and_addresses, hr_1, college_3, soccer_2, scientist_1, products_gen_characteristics, program_share, swimming, manufacturer, school_bus, cre_Doc_Tracking_DB, book_2, performance_attendance, music_2, soccer_1, customer_deliveries, document_management, insurance_fnol, wine_1, company_office, theme_gallery, county_public_safety, medicine_enzyme_interaction, sakila_1

Table A.2: Development sample databases for Sample 1 and Sample 2

Set	Sample 1	Sample 2
Dev	orchestra, concert_singer, tvshow, museum_visit	orchestra, concert_singer, tvshow, museum_visit, flight_2, pets_1, world_1, poker_player, battle_death, wta_1, cre_Doc_Template_Mgt, singer, employee_hire_evaluation, network_1, course_teach, real_estate_properties, voter_1, student_transcripts_tracking, dog_kennels

Table A.3: Train sample databases for Sample 3 and Sample 4

Set	Sample 3	Sample 4
Train	architecture, farm, small_bank_1, apartment_rentals, entertainment_awards, restaurant_1, pilot_record, cinema, climbing, railway, twitter_1, journal_committee, musical, wedding, riding_club, customers_and_invoices, department_management, city_record, shop_membership, perpetrator, company_1, ship_1, game_injury, chinook_1, entrepreneur, gymnast, phone_market, election_representative, formula_1, phone_1, driving_school, music_4, baseball_1, university_basketball, e_learning, sports_competition, assets_maintenance, station_weather, culture_company, school_finance, film_rank, party_host, ship_mission, company_employee, tracking_share_transactions, news_report, storm_record, gas_company, solvency_ii, cre_Drama_Workshop_Groups, cre_Theme_park, candidate_poll, tracking_orders, tracking_software_problems, student_assessment, wrestler, local_govt_and_lot, product_catalog, behavior_monitoring, e_government, products_for_hire	architecture, farm, small_bank_1, apartment_rentals, entertainment_awards, restaurant_1, pilot_record, cinema, climbing, railway, twitter_1, journal_committee, musical, wedding, riding_club, department_store, mountain_photos, insurance_policies, train_station, club_1, debate, flight_company, election, loan_1, flight_1, party_people, school_player, store_product, network_2, browser_web, roller_coaster, dorm_1, cre_Doc_Control_Systems, customers_and_products_contacts, student_1, voter_2, movie_1, epinions_1, customer_complaints, insurance_and_eClaims, aircraft, cre_Docs_and_Epenses, manufactory_1, flight_4, store_1, protein_institute, coffee_shop, customers_campaigns_ecommerce, local_govt_mdm, csu_1, activity_1, workshop_paper, race_track, match_season, music_1, machine_repair, tracking_grants_for_research, icfp_1, customers_card_transactions, local_govt_in_alabama, body_builder, bike_1, railway, device

Table A.4: Development sample databases for Sample 3 and Sample 4

Set	Sample 3	Sample 4
Dev	orchestra, concert_singer, tvshow, museum_visit, flight_2, pets_1, world_1, poker_player, battle_death, wta_1, cre_Doc_Template_Mgt, singer, employee_hire_evaluation, network_1, course_teach, real_estate_properties, voter_1, student_transcripts_tracking, dog_kennels	orchestra, concert_singer, tvshow, museum_visit, flight_2, pets_1, world_1, poker_player, battle_death, wta_1, cre_Doc_Template_Mgt, singer, employee_hire_evaluation, network_1, course_teach, real_estate_properties, voter_1, student_transcripts_tracking, dog_kennels

Appendix B

Confusion Matrices

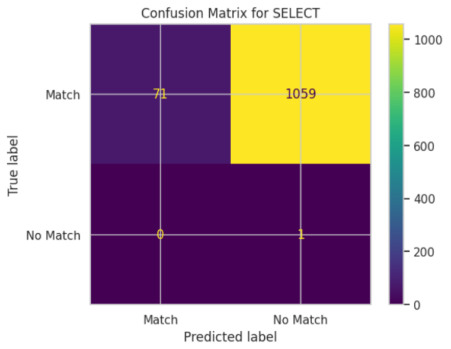

Figure B.1: Confusion Matrix for *SELECT* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 1

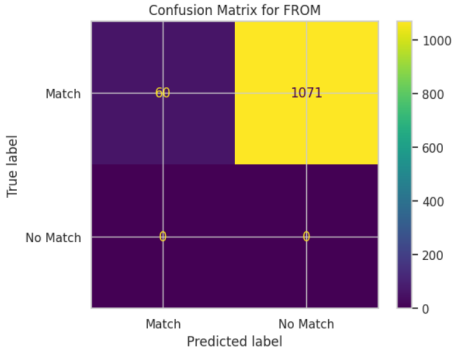

Figure B.2: Confusion Matrix for *FROM* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 1

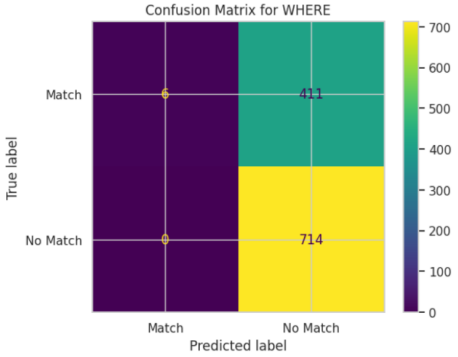

Figure B.3: Confusion Matrix for *WHERE* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 1

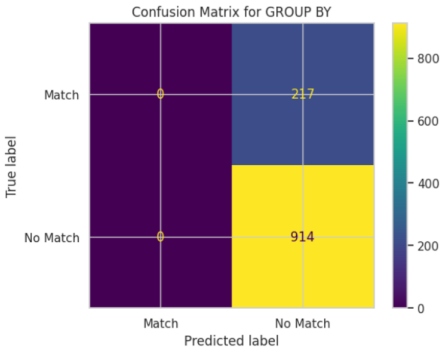

Figure B.4: Confusion Matrix for *GROUP BY* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 1

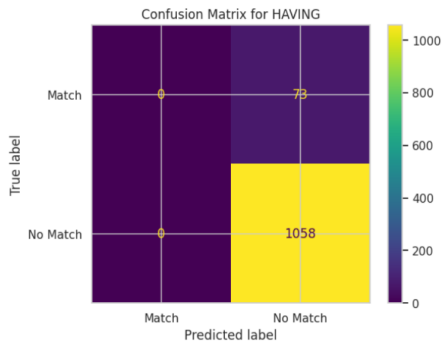

Figure B.5: Confusion Matrix for *HAVING* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 1

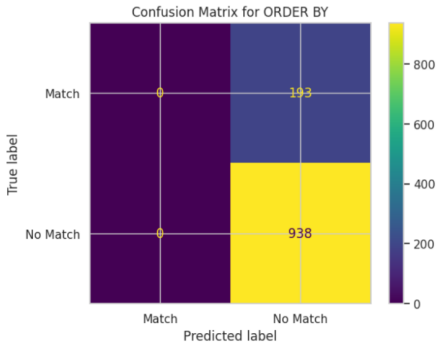

Figure B.6: Confusion Matrix for *ORDER BY* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 1

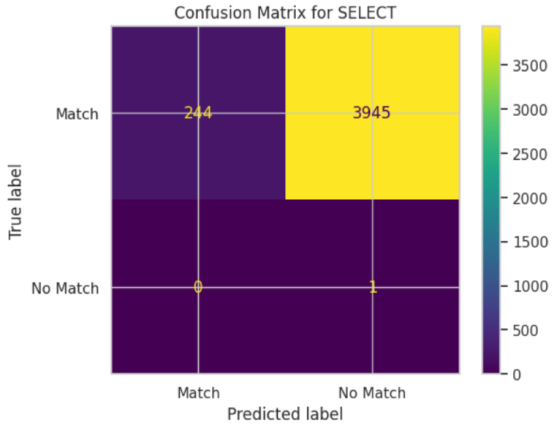

Figure B.7: Confusion Matrix for *SELECT* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 2

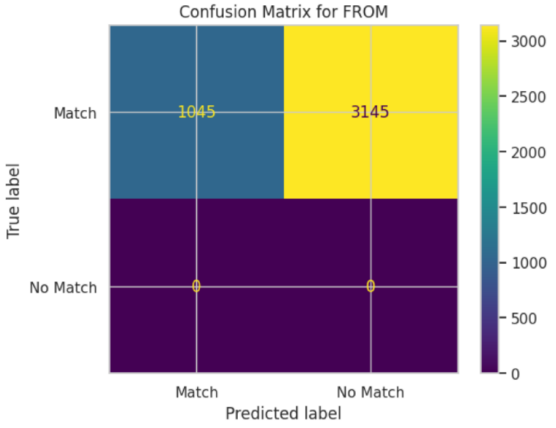

Figure B.8: Confusion Matrix for *FROM* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 2

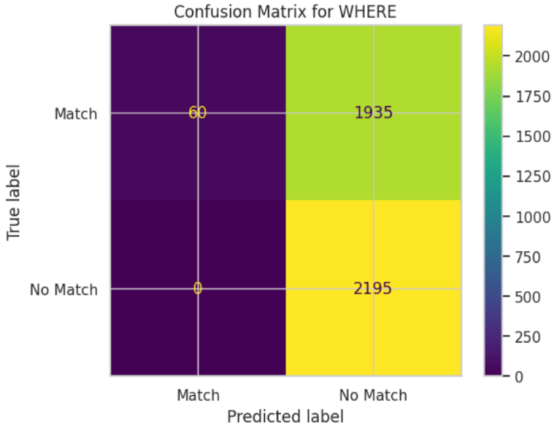

Figure B.9: Confusion Matrix for *WHERE* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 2

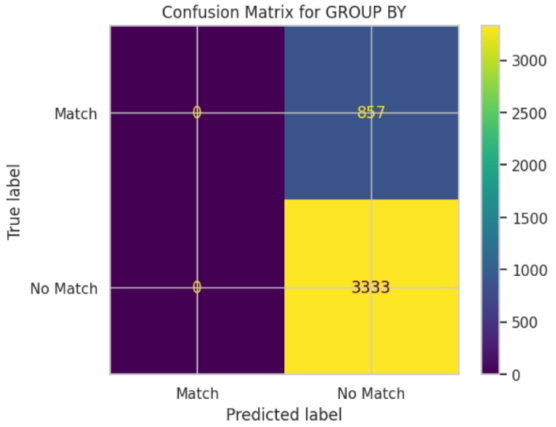

Figure B.10: Confusion Matrix for *GROUP BY* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 2

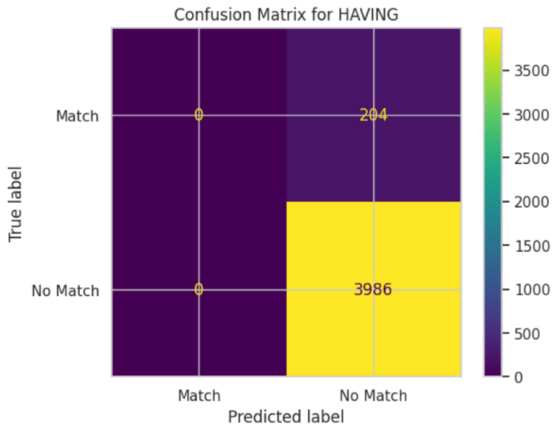

Figure B.11: Confusion Matrix for *HAVING* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 2

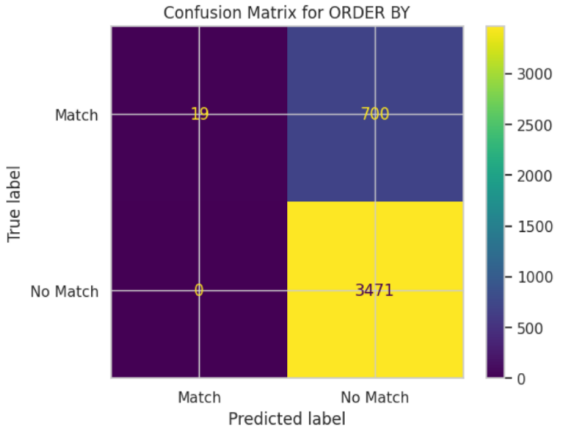

Figure B.12: Confusion Matrix for *ORDER BY* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 2

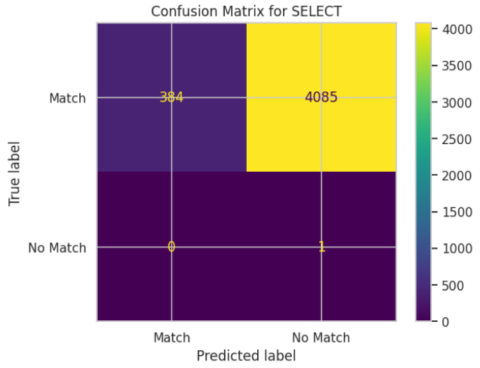

Figure B.13: Confusion Matrix for *SELECT* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 3

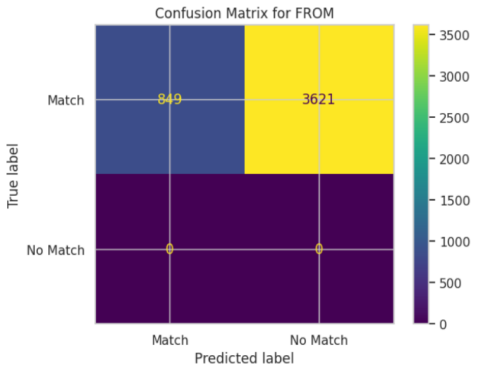

Figure B.14: Confusion Matrix for *FROM* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 3

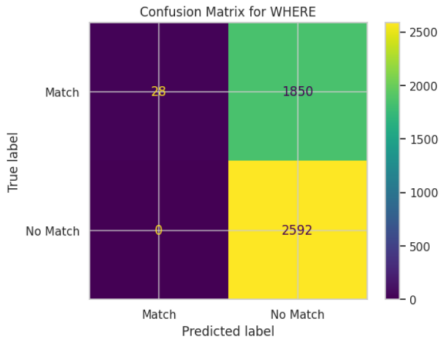

Figure B.15: Confusion Matrix for *WHERE* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 3

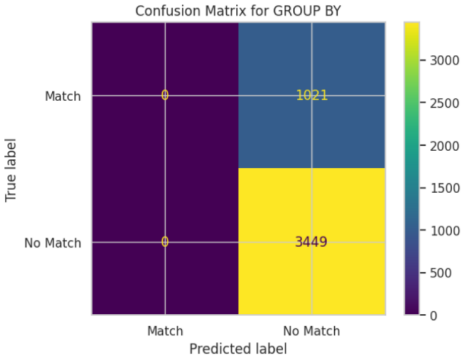

Figure B.16: Confusion Matrix for *GROUP BY* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 3

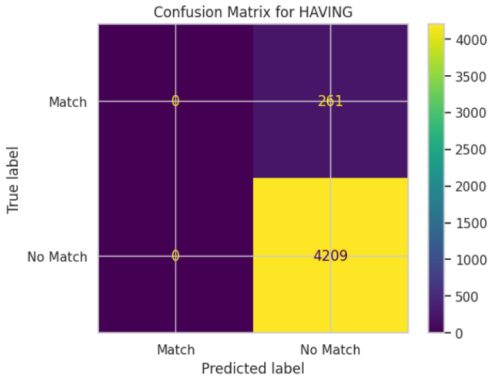

Figure B.17: Confusion Matrix for *HAVING* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 3

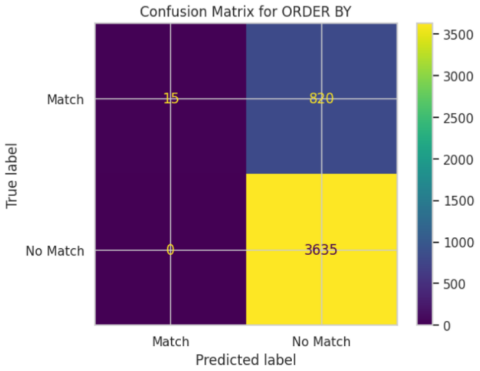

Figure B.18: Confusion Matrix for *ORDER BY* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 3

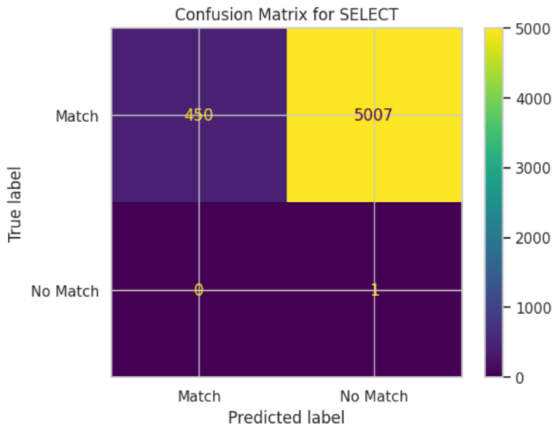

Figure B.19: Confusion Matrix for *SELECT* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 4

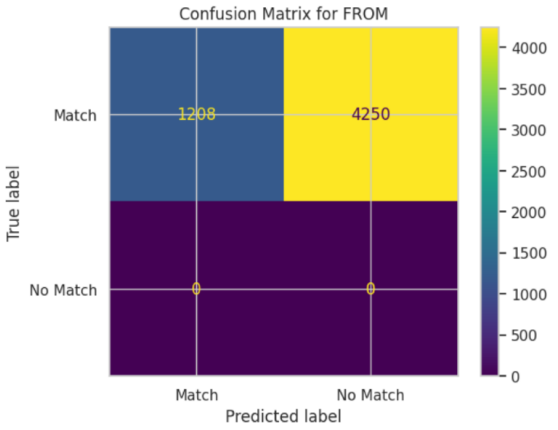

Figure B.20: Confusion Matrix for *FROM* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 4

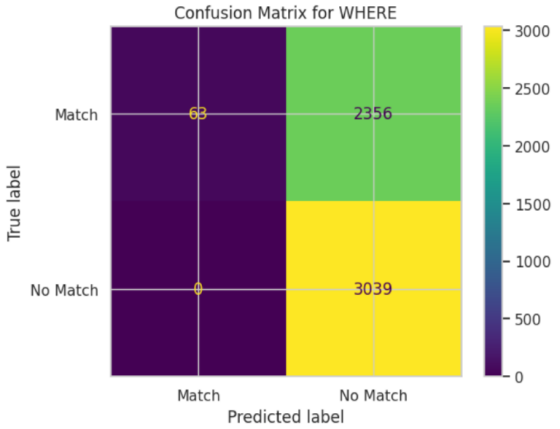

Figure B.21: Confusion Matrix for *WHERE* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 4

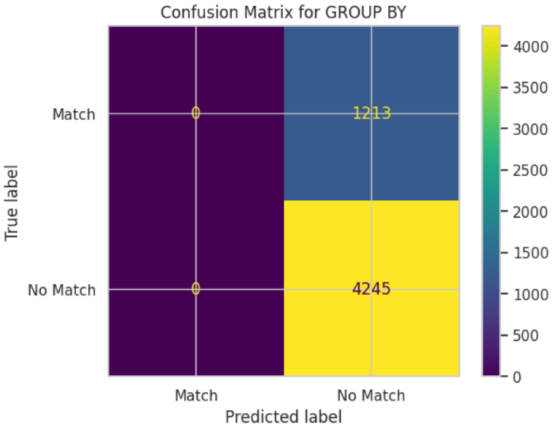

Figure B.22: Confusion Matrix for *GROUP BY* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 4

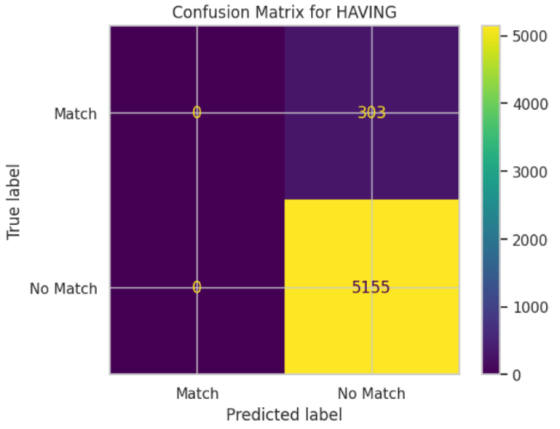

Figure B.23: Confusion Matrix for *HAVING* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 4

Figure B.24: Confusion Matrix for *ORDER BY* clause regarding results from the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 4

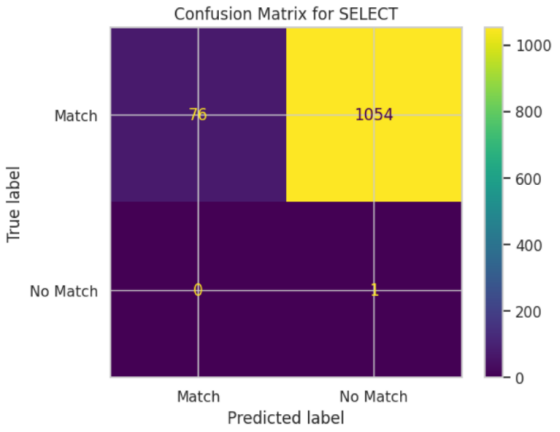

Figure B.25: Confusion Matrix for *SELECT* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 1

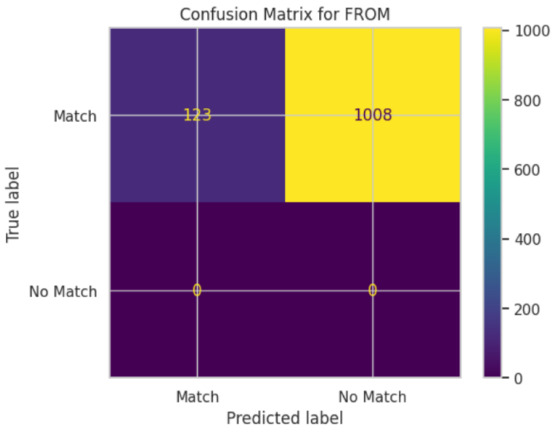

Figure B.26: Confusion Matrix for *FROM* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 1

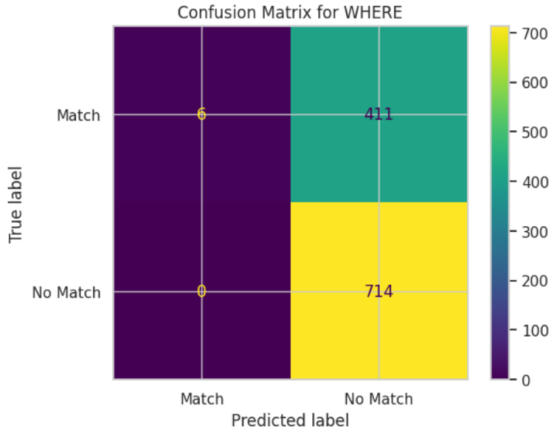

Figure B.27: Confusion Matrix for *WHERE* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 1

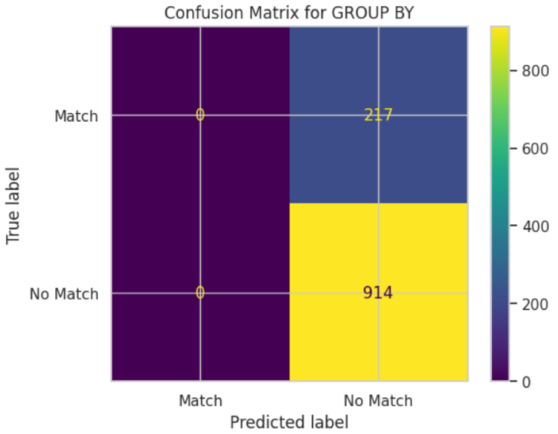

Figure B.28: Confusion Matrix for *GROUP BY* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 1

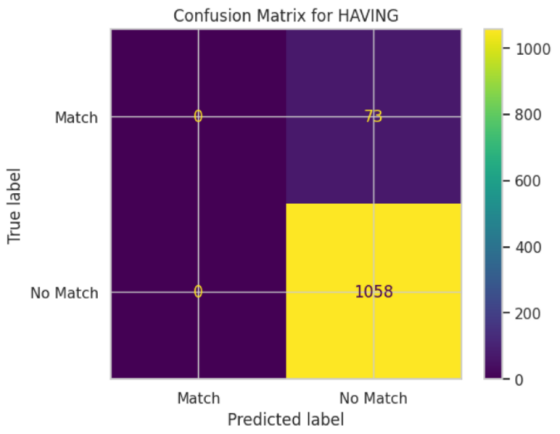

Figure B.29: Confusion Matrix for *HAVING* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 1

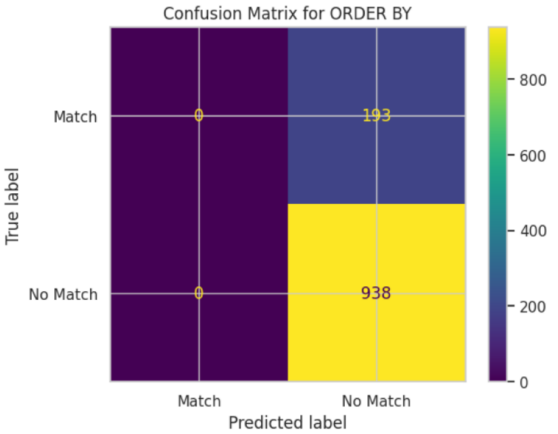

Figure B.30: Confusion Matrix for *ORDER BY* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 1

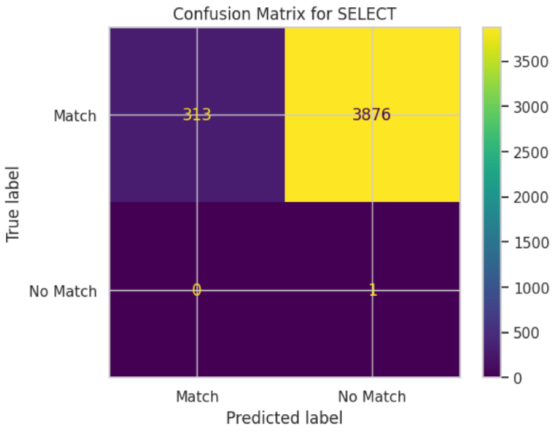

Figure B.31: Confusion Matrix for *SELECT* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 2

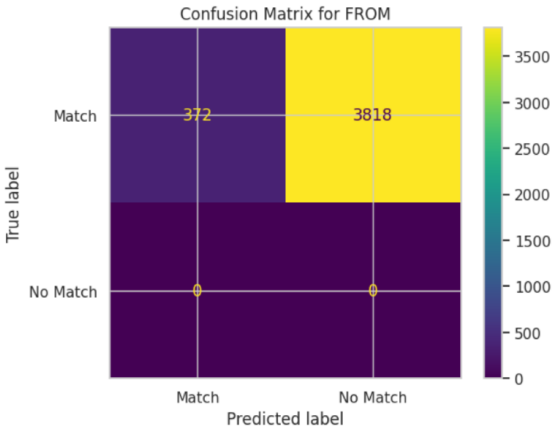

Figure B.32: Confusion Matrix for *FROM* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 2

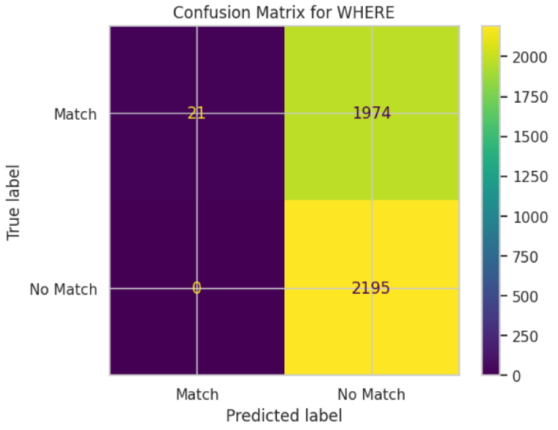

Figure B.33: Confusion Matrix for *WHERE* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 2

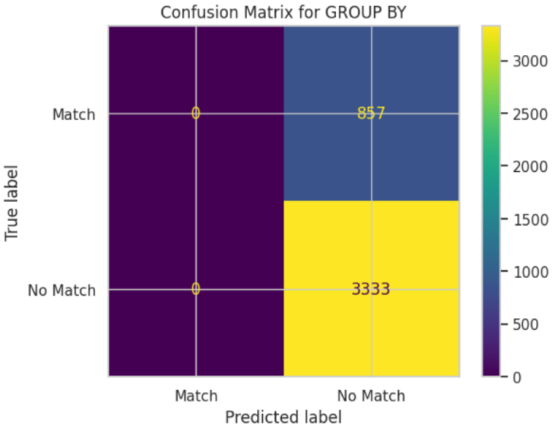

Figure B.34: Confusion Matrix for *GROUP BY* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 2

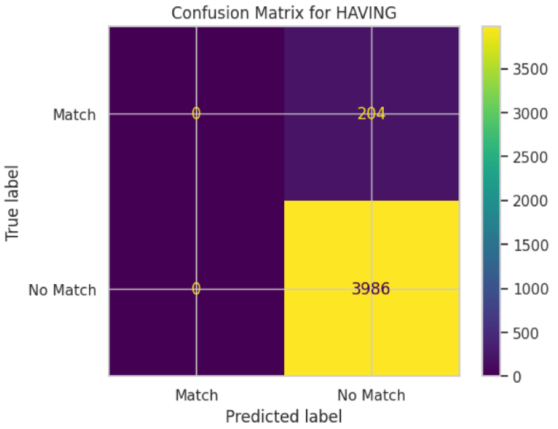

Figure B.35: Confusion Matrix for *HAVING* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 2

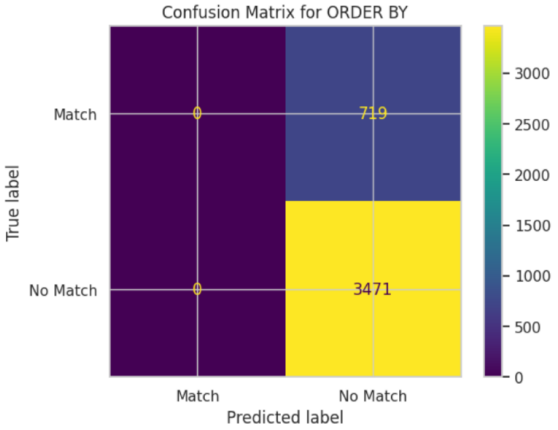

Figure B.36: Confusion Matrix for *ORDER BY* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 2

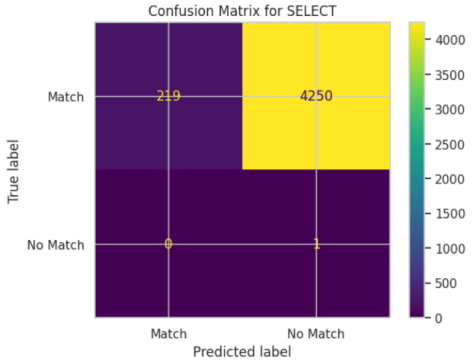

Figure B.37: Confusion Matrix for *SELECT* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 3

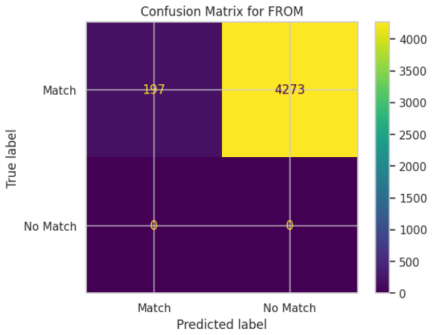

Figure B.38: Confusion Matrix for *FROM* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 3

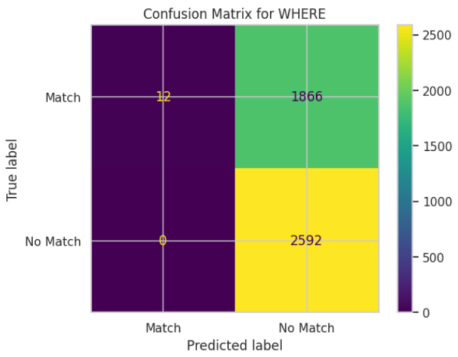

Figure B.39: Confusion Matrix for *WHERE* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 3

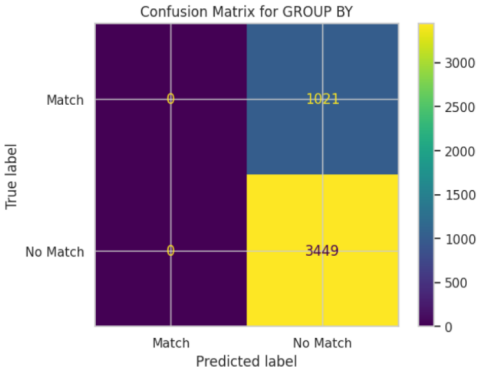

Figure B.40: Confusion Matrix for *GROUP BY* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 3

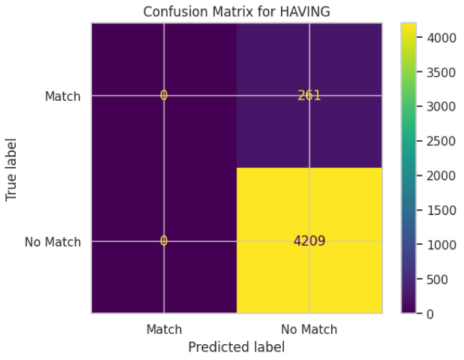

Figure B.41: Confusion Matrix for *HAVING* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 3

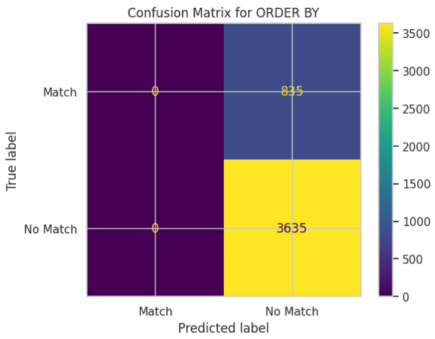

Figure B.42: Confusion Matrix for *ORDER BY* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 3

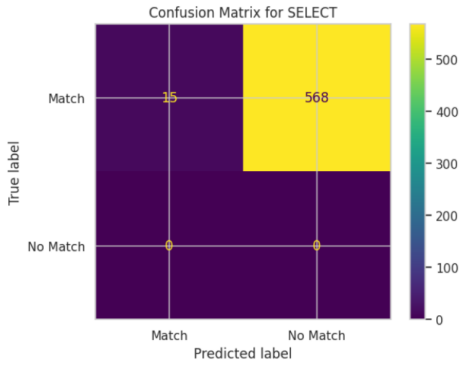

Figure B.43: Confusion Matrix for *SELECT* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 4

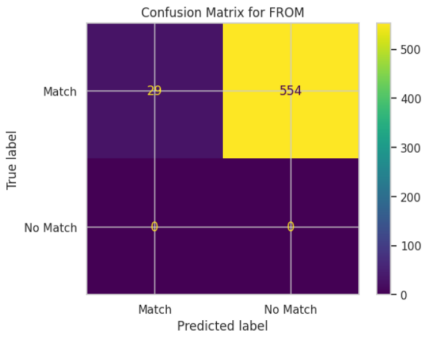

Figure B.44: Confusion Matrix for *FROM* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 4

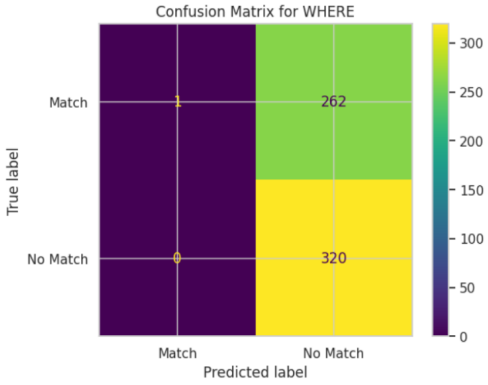

Figure B.45: Confusion Matrix for *WHERE* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 4

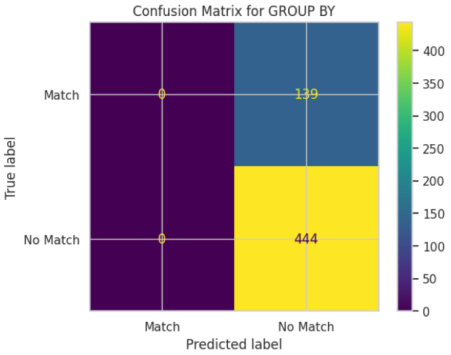

Figure B.46: Confusion Matrix for *GROUP BY* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 4

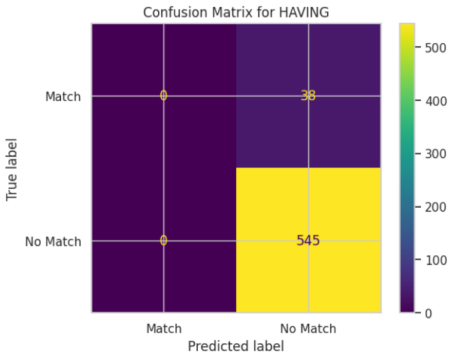

Figure B.47: Confusion Matrix for *HAVING* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 4

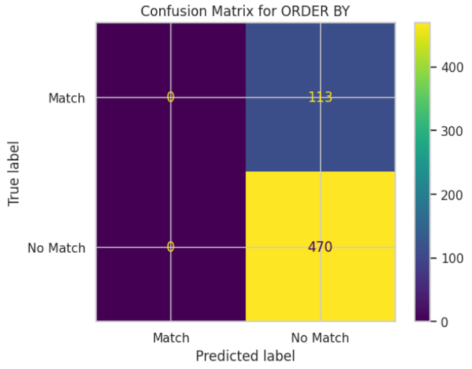

Figure B.48: Confusion Matrix for *ORDER BY* clause regarding results from training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 4

Appendix C

Training and validation loss graph results

Figure C.1: Train and validation loss results regarding the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 1

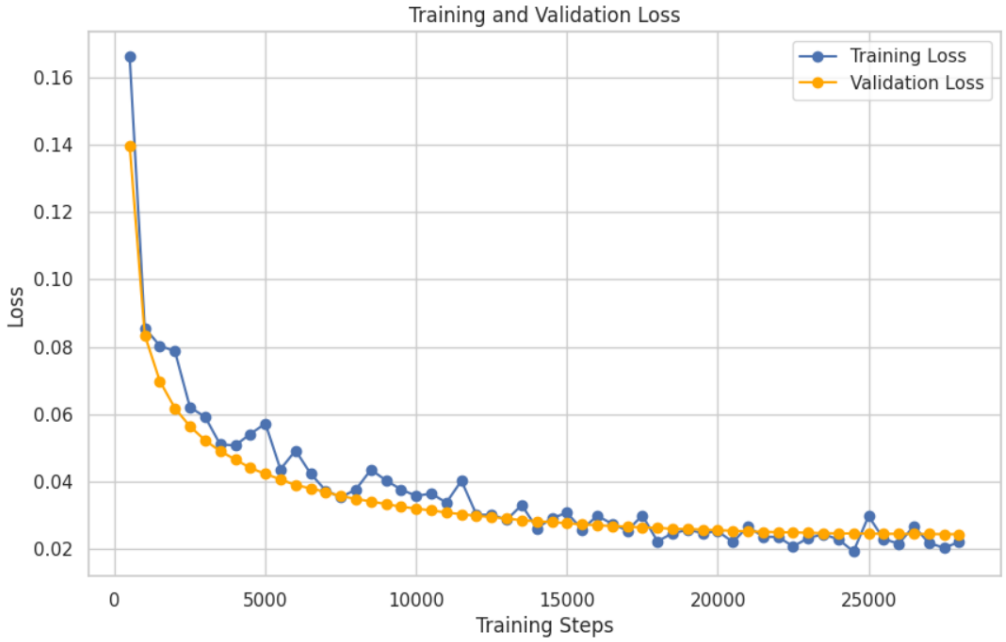

Figure C.2: Train and validation loss results regarding the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 2

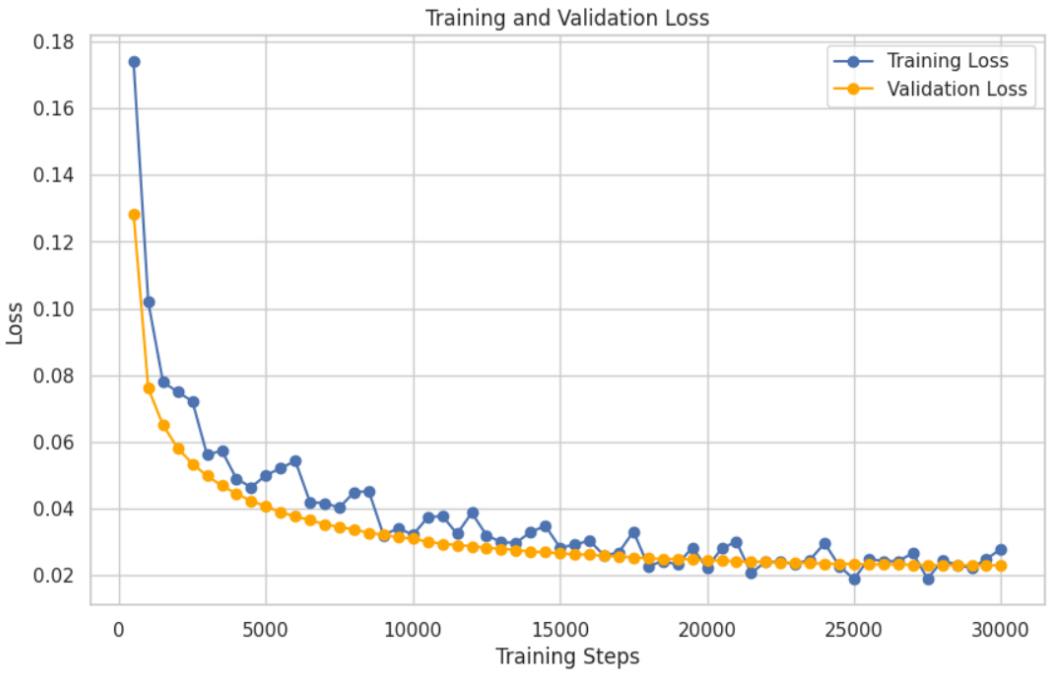

Figure C.3: Train and validation loss results regarding the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 3

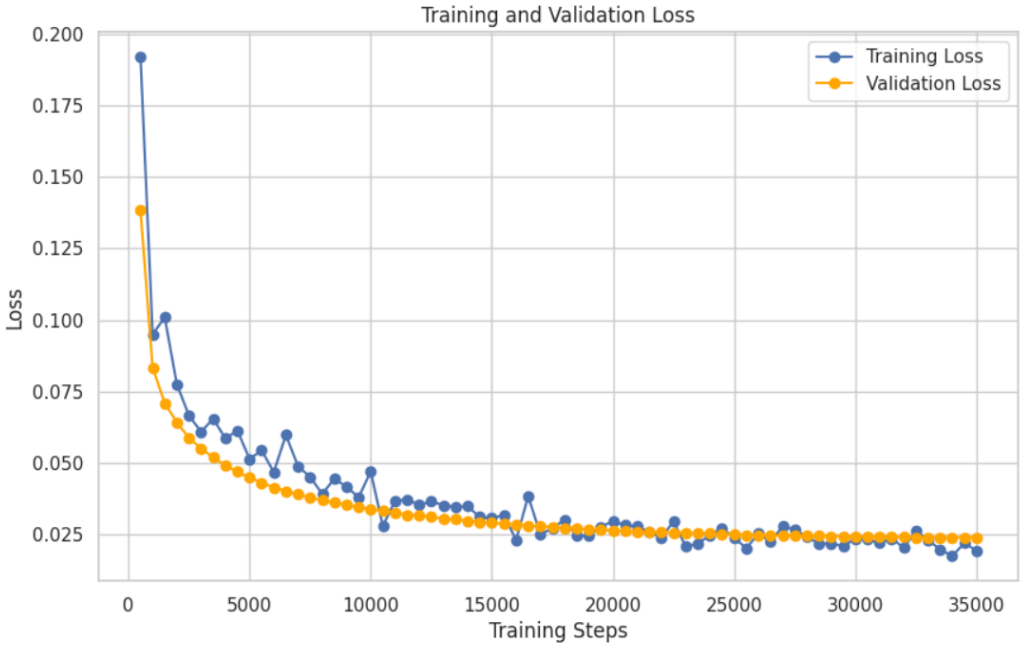

Figure C.4: Train and validation loss results regarding the pre-training in T5-base model with *WikiSQL* data later fine-tuned with *SParC* data for sample 4

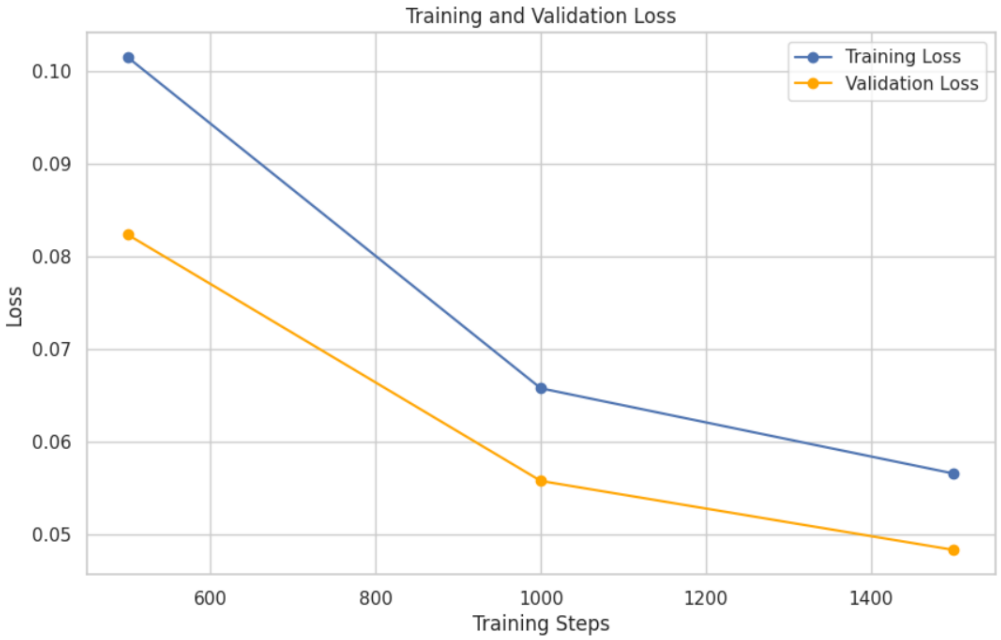

Figure C.5: Train and validation loss results regarding training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 1

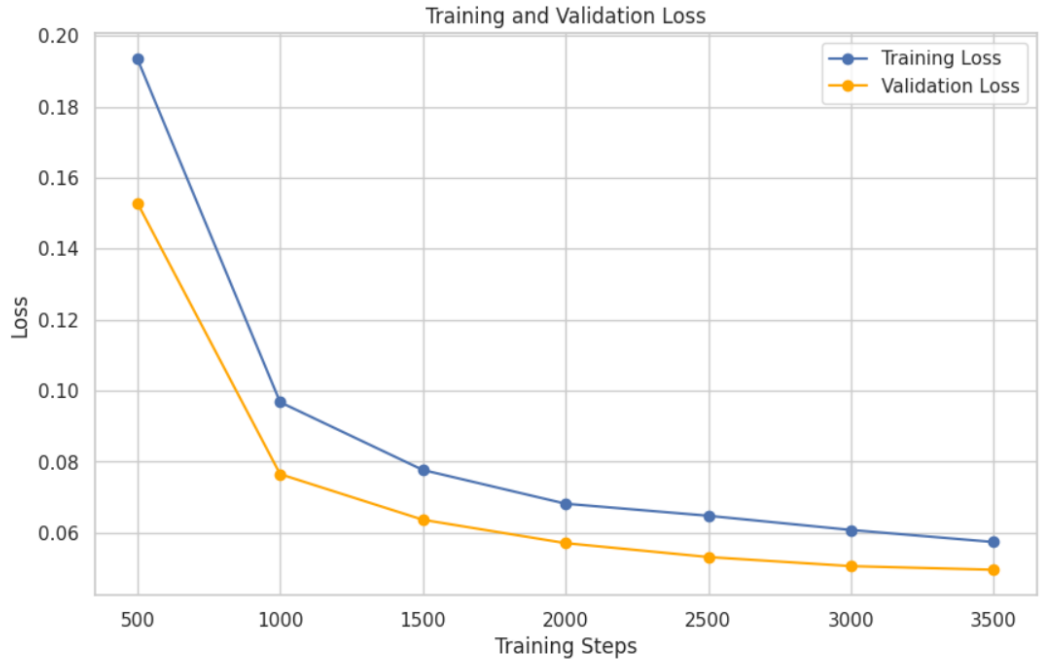

Figure C.6: Train and validation loss results regarding training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 2

Figure C.7: Train and validation loss results regarding training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 3

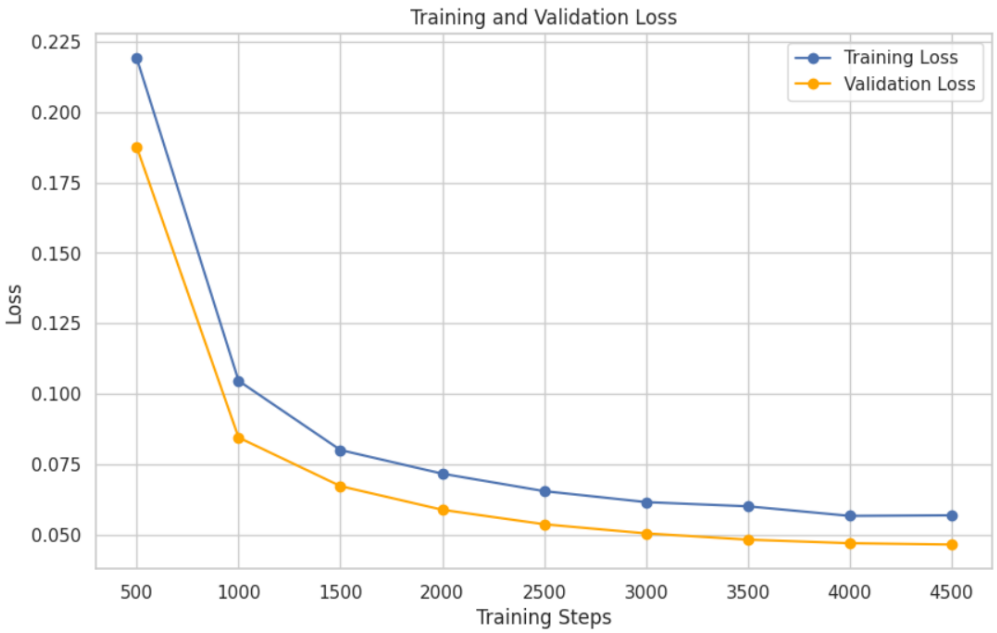

Figure C.8: Train and validation loss results regarding training *SParC* data with a pre-train in T5-base model with fine-tuning for sample 4

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

Universidade Nova de Lisboa