TimeSeq:
Sequences with Normalized Time Information for Seq2Seq Translation Task

to be defended on June 2021

Aram Javier Villasana Falcón
Author

Marta Ruiz Costa-Jussà
Department of Computer Science
Advisor

Meritxell Gonzalez Bermudez
Oxford University Press
Co-Advisor

MASTER IN ARTIFICIAL INTELLIGENCE
FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

ABSTRACT

During last years, researchers have focused on developing Deep Learning models to translate natural language questions to SQL structured queries, what is known as the “text-to-sql” translation task. By 2020, DL models have reached outstanding results in WikiSQL and Spider challenges. However, there are still some challenges they have not addressed. **Time normalization** and **small size of training samples** stand out among these challenges. Nowadays, state-of-the-art models for text-to-sql task are not able to deal with common questions like, “what were the average sales for last three Christmas?” or “how much did I sell in every weekend of the last quarter of previous year?”. On the other hand, models are trained with relatively small number of samples compared to the number of parameters in the deep learning models and collecting more questions and annotate the dataset is not easy. For these reasons, we propose **TimeSeq**, a new standard for annotating temporal information as normalized token sequences (**time sequences**), which can take advantage of DL models that perform seq-to-seq translation for automatically normalizing relative time information contained in common questions about business databases. Currently this standard applies to the context of questions made about the content of a transactional database; but could grow to other domains. We demonstrated with our experiments that deep learning models can learn to summarize the temporal information contained in a natural language question into a time sequence. We developed a process for **data augmentation to increase the examples of pairs of questions and time sequences**, **based in the substitution of paraphrases** corresponding to 3 ontologies we integrated: (1) **Ontology of Paraphrases for Temporal Expressions**, (2) **Ontology of Paraphrases for General Business Database Querying**, and (3) Ontology of Paraphrases for Specific Industry Domain, and an algorithm to perform the data augmentation, by creating new pairs of questions and time sequences without affecting the fidelity of the temporal information, as result of controlled combinations of paraphrases substituted at original pairs and at the new pairs generated. Our experiments also demonstrate the usefulness of paraphrases substitution for providing data to train models in the translation from questions to time sequences, that can generalize part of the learning to unseen datasets (0.65 accuracy in validation and test sets vs. 0.48 accuracy when training without data augmentation). However, training variability is not enough to represent the whole variability of unseen data, i.e., the models generalize what they can with the variability present in the data they were trained. Although our data augmentation method based in paraphrases substitution still needs to improve, it provides a relevant contribution for developing new data augmentation methods that enable the use of DL models in cases where the number of examples is too few for training.

CONTENTS

Table of Contents

Chapter 1

1. INTRODUCTION

1.1. Motivation

Nowadays, most cognitive assistants are focused towards answering questions of users about non-structured data, trained with vast amounts of Q&A pairs and conversation logs, and enriched with domain relevant information for learning how to deliver answers in the most human-like possible way. However, there are still few examples of cognitive assistants trained to respond natural language questions which require a precise answer based on to the content of transactional databases, capable of learning by themselves the correct tables and columns to query, as well as the query structures which correspond to the natural language question. Although, this capability of being able to query structured data, could seems less challenging that being able to get complex answers from non-structured data; it offers a pragmatic value for businesses and users of other domains which require rapid and exact answers from the transactional data they have.

Most of the time, final users who require data from transactional databases cannot get immediate answers, since they depend on data analysts for querying the data or interpret the data obtained from the databases. Although some final users have BI tools which offer a more friendly environment to visualize data in real time, they struggle to find the exact data they need, because they are not experts about how the data is distributed across the database tables and the exact names of the fields they need. For these reasons, a cognitive assistant who could deliver immediate and precise answers to final users about the structured data they have, would really empower them to achieve more efficient and effective outcomes.

It is complicated to effectively handle the ambiguity of natural language questions in the domain of transactional database querying. Besides identifying the correct table and column names and values, the ambiguity affects the definition of the correct query structure (SQL). During last years, researchers have focused to develop Deep Learning models for translating natural language questions into SQL structured queries, what is known as the “text-to-sql” translation task[1]. However, there are challenges that have not been yet addressed by state-of-the-art models for this task[1]. Our intention with this work is to develop proposals that contribute to reach an effective solution to some of these challenges.

[1] (Zhong et al, 2017) and (Yu et al, 2019)

Additionally, there is few research about the text-to-sql translation in Spanish language: only Gámez et al. (2013) was found in literature review. For this reason, we want to contribute with our work to expand the research about this task in Spanish language.

1.2. The Problem Statement

Current state-of-the-art models solving the text-to-sql task, are not able to deal with common questions like, “what were the average sales for last three Christmas?” or “how much did I sell in every weekend of the last quarter of previous year?” For being able to translate these expressions into SQL query structure, it is required to perform a time normalization, which implies the ability to address the specific periods of time and points in the time the natural language question is denoting, in relation to the actual time in which the question is being expressed.

Deep learning models deal with the text-to-sql task by separating the SQL structure into distinct slots that are filled by using sequence-to-sequence translation from a question to a specific sequence corresponding to each slot (Zhong et al, 2017). It would be useful if time constraints could be handled with same approach, i.e., by using a sequence-to-sequence translation from a question to a normalized sequence containing the temporal constraints contained in the question. For this reason, we propose a set of rules for annotating temporal information contained in questions into sequences containing such information in a structured way that allows to correctly perform the queries constrained by temporal information.

Thus, the **first research problem** we face is: *“Can deep learning models learn to translate natural language questions into normalized sequences that summarize the complete temporal constraints?”.*

However, there is many combinations of temporal expressions and non-temporal tokens that we can find in natural language questions about the content of a transactional database, and there was not found any public database of questions in Spanish containing relative time expressions. Collecting enough instances to represent the whole variability of such combinations for training deep learning models would be a hard job. We explored for options to integrate a database in Spanish language, containing question with absolute and relative temporal constraints. Among best options we found that databases of paraphrases are a common resource when trying to generate variability for given natural language expressions (Ganitkevitch & Callison-Burch, 2014).

This way, we decided to integrate our own database by gathering a few hundred questions and using paraphrasing to explode questions to hundred thousand or millions. Here is where our **second research problem** emerges: *"Does Data Augmentation performed through direct substitution of paraphrases can help deep learning models to learn how to translate natural language questions into time sequences?".*

1.3. Research Objectives

Considering the two problems stated, the following research objectives are defined:

Problem 1: *"Can deep learning models learn to translate natural language questions into normalized sequences that summarize the complete temporal constraints?".*

- **Research Objective 1.1.** Develop a set of rules to annotate the temporal information contained in natural language questions as temporal normalized sequences which contain all temporal constraints necessary to perform a correct search, in the context of questions made about the content of a transactional database.

- **Research Objective 1.2.** Evaluate if deep learning models can learn to summarize the temporal information contained in a natural language question into a normalized sequence.

Problem 2: *"Does Data Augmentation performed through direct substitution of paraphrases can help ML/DL models to learn how to translate natural language questions into time sequences?".*

- **Research Objective 2.1.** Develop a method for performing data augmentation, capable of covering wide variability of combinations of temporal expressions and other frequent tokens in natural language questions, in the context of questions made about the content of a transactional database.

- **Research Objective 2.2.** Develop an algorithm to efficiently perform the data augmentation process.

- **Research Objective 2.3.** Build a model for sequence-to-sequence translation, capable of performing the translation from natural language questions containing temporal information into temporal normalized sequences, in the context of questions made about the content of a transactional database that were not used to train the model.

- **Research Objective 2.4.** Evaluate if models trained with this Data Augmentation method can generalize to unseen datasets of questions.

1.4. Scope of Analysis and Application

For conducting this research, it has been selected a specific Business Domain with data about sales of products in Spanish Language, as described in the chapter 5: “Dataset Integration”. However, the contributions derived from this work can be perfectly extended to other domains which require to perform time normalization in order to complete a translation from natural language questions into precise SQL query structures for extracting accurate information from transactional databases.

Important to note that the scope of application of the temporal normalized sequences, the data augmentation method, and the approach for translating natural languages questions to these temporal normalized sequences is limited to natural language questions asked for querying precise content from a transactional database, since these contributions are taking advantage of the small size and simplicity these questions exhibit in comparison with other more complex contexts. Moreover, the set of rules for annotating the temporal normalized sequences corresponding to the natural language questions, is specifically designed for handling questions that fits the “basic structure” of an SQL query, as discussed in chapter 4: “Description of Domain”. Although basic, this structure is relevant because more complex questions and their corresponding SQL queries uses this basic structure as base component of their own structure. With this basic structure, it can be answered any question asking about the sum, average, minimum or maximum of the values accumulated during certain historic period or periods, for any aggregation of columns in the table, like for example: “what were the average sales for last three Christmas by brand and product category?” or “how much did I sell in every weekend of the last quarter of previous year in each region and store?”.

When applying the contributions of this work to other application domains inside the described scope, some adaptations of the used ontologies could be required. For applying the contributions of this work in other languages, an adaptation of used ontologies would be necessary. It is important to note that time normalization performed during this work is limited to handling dates, without including time (hours, minutes, seconds, etc.) information.

Chapter 2

2. LITERATURE REVIEW

2.1. Text-to-SQL Task

During last years, researchers have focused to develop Deep Learning models for translating natural language questions to SQL structured queries, what is known as the "text-to-sql" translation task. By 2020, DL models have reached outstanding results (in WikiSQL and Spider challenges)[2]. However, there are still some challenges they have not addressed:

- **Ambiguity in the names of values, columns and tables expressed in natural language questions.** For the main training datasets of questions (WikiSQL and Spyder), most of the time the tables, columns and values are named with the same name they appear in the provided Database for querying.

- **Training Sample Size.** Models are trained with relatively small number of samples compared to the number of parameters in the deep learning models. WikiSQL has only 7,000 questions, while Spider has 8,659 questions. On the other hand, getting a new txt-to-sql dataset for training and testing deep learning models is a complicated task because the effort it requires to collect the questions and annotate the dataset.

- **Time Normalization.** WikiSQL and Spider datasets do not provide questions containing relative time expressions. This way, state-of-the-art models for text-to-sql task are not able to deal with common questions like, "what were the average sales for last three Christmas?" or "how much did I sell in every weekend of the last quarter of previous year?"

- **Diversity of Languages.** Models have been trained exclusively in English Language datasets.

2.2. Time Normalization

Time normalization implies the ability to address specific points or periods in a standard timeline, when they are referred in relation to the actual time or other temporal reference appearing in a natural language expression. Popular schemes used for annotating time expressions existing in

[2] WikiSQL (Zhong et al., 2017), Spider (Yu et al., 2019)

natural language sentences are the well-known TIDES (Ferro et al., 2005), ISO-TimeML (Pustejovsky et al., 2010) and, more recently, SCATE (Bethard and Parker, 2016).

Figure 1.1, below, illustrates the difference between SCATE representation of temporal information vs. the normalized time sequences proposed in this work (TimeSeq), which will be explained in detail in Chapter 6.

SCATE representation

```
ThisRIs(
  Between(
    LastRI(
      today,
      Intersection(Set(
        RepeatingField(MONTH_OF_YEAR, 3),
        RepeatingField(DAY_OF_MONTH, 6)))),
    today),
    RepeatingField(DAY_OF_WEEK, 6))
```

TimeSeq representation

EVERY 06_SABADO FROM LAST 01_MARZO 16

Figure 1.1: *SCATE and TimeSeq representations for expression "every Saturday since March 6".*

Chapter 3

3. DESCRIPTION OF DOMAIN

3.1. Introduction to the Domain

As explained before, this work focuses on natural language questions containing temporal information and asking about precise data contained in a transactional database in the Business domain, specifically data about sales of products. It has been argued that the scope of application of the proposed temporal normalized sequences, the data augmentation method, and the approach for translating natural languages questions to these temporal normalized sequences take advantage of the small size and simplicity this kind of questions exhibits in comparison with other natural language expressions in more complex domains. Additionally, the set of rules for annotating the temporal normalized sequences corresponding to the natural language questions, is specifically designed for handling questions that fits a “basic structure” of an SQL query, which is the base structure for more complex queries. The goal of this chapter is to introduce the basic SQL query structure this work is focusing on, and to provide a detailed explanation of the kind of questions that can be answered with this basic query structure. With a clear understanding of the kind of questions this work is tackling, the chapter proceeds to explain the difficulties emerged from these questions which denotes the necessity of using Time Normalization.

3.2. Understanding of Basic SQL Query Structure

For the purposes of this work, a basic SQL structure is composed by four sections, as observed in Figure 3.1 below:

Figure 3.1: *Basic SQL query structure, corresponding to question: "How much carbonated drinks of Kas brand the company sold every weekend during the first quarter of the year at the North store?".*

The four sections contained in this structure are:

1. **Request ("Select"):** Declaration of the information the user wants to visualize. There are two possible scenarios about the content of requested information. In the first scenario, requested information can include specific declared fields for which the user wants to get all the values stored in certain records, which meet specific constraints. For example, if a user requests: "all sales by store, category, brand and date", the request section would be encoded as "Select Store, Category, Brand, Sale_Date, Sale_Value".

   In the second scenario, requested information can include the result of calculating "aggregate function(s)"[3] performed over certain field(s) for the records that meet the imposed constraints, and it can be calculated for the total set of such records or it can be calculated for sub-sets of these records, where each sub-set is an existing combination of the values of the distinct fields requested that are not target of an aggregate function. For

[3] An aggregate function performs a calculation (such as sum, average, minimum, maximum) on a set of values, and returns a single value.

example, if a user requests: “total monthly sales by brand for all years”, the request section would be encoded as “Select Brand, Year, Month, Sum(Sale_Value)”.

2. **Source (“From”):** Table (or tables) containing the requested information. For the case of this work, this section is not relevant, since the task is considering only temporal information, and it is supposed that temporal information is not affecting the selection of source tables.

3. **Constraints (“Where”):** All possible conditions constraining the query. Conditions can involve time conditions or not.

4. **Aggregation Level (“Group by”):** This section gets active only when the user makes an information request that implies to use and aggregate function calculated for distinct sub-sets of the extracted records, where each sub-set is an existing combination of the values of the distinct fields requested that are not target of the aggregate function (second scenario detailed in the description of the “Request” section). For example, the question “how much the company sold for each brand every month during last 3 years?” would imply the sub-sets integrated by the combination of the values of the fields brand, year and month, which would be encoded as “Group by Brand, Year, Month”.

Since this work is based exclusively in this basic SQL query structure, the kind of questions that are handled by this work are constrained to the question that this structure can answer. Such questions will be detailed in following section.

3.3. Understanding of Questions in the Domain of Study

Considering this work is focused on handling questions that fits the “basic structure” of an SQL query, it is important to characterize the class of questions that can be answered with this basic query structure, to discard those questions that are not in the domain of study and application of this work.

In chapter 1, it was stated that with this basic structure, it can be answered ***any question asking about the sum, average, minimum or maximum of the values accumulated during certain historic period or periods***, for any aggregation of columns in the table, like for example: “what were the average sales for last three Christmas by brand and product category?” or “how much did I sell in every weekend of the last quarter of previous year in each region and store?”. Observing in detail the previous description about the type of questions that the basic SQL query structure is able

to respond, we observed that such questions must be based in historic records ("*historic period or periods"*) and they request the calculation of an indicator ("*sum, average, minimum or maximum"*) that can be extracted directly from historic values in a single operation, as in opposition to possible question: "what is the percentage of year sales that are sold during Christmas season, considering last three years", which clearly implies to perform the whole query using three operations or stages: firstly, calculating the sum of the sales for las three Christmas; secondly, calculating the sum of the sales for last three years; and lastly, dividing the value of the first sum by the value of the last sum. Finally, we observe the questions inside the scope are asking for the indicator by itself, they are not asking to compare or consider the relative position of elements mentioned in question with regards to the value of the calculated indicator, as in opposition to the possible question: "what were the 5 product categories with the largest average sales for last three Christmas?", which would imply to calculate the indicator (average of sales for last three Christmas) for every product category, and then, sorting the categories by descending average value and keeping top 5 categories.

Therefore, it is observed that questions can be characterized, depending on the characteristics of the answer they are expecting to receive:

1) **Historic vs. Future Values:** Identifies if a question is asking about past values (historic) or if is asking about prediction of future values, based in historic values (forecast). An example of a question asking for historic values is "how much did I sell the last December?". An example of question asking about future values is "how much can I expect to sell next December?".

2) **Simple vs. Complex Values:** Identifies if a question can be answered with a value obtained directly from one operator applied to an aggregated group of records (simple, e.g., sum, average, maximum, minimum), or if it requires to perform more than one operation before delivering the requested value (complex value, e.g., proportion or rate). An example of a question asking for simple values is "how much were the sales by brand during last quarter?". An example of question asking about complex values is "what was growth rate for sales of last quarter compared to one year ago, for each brand?".

3) **Direct Value vs. Value Derived:** Identifies if a question is asking for a specific value or if it is asking to perform any kind of operation taking into account the relative position of elements mentioned in the question, derived from the value calculated for each element (e.g., "top N elements", "last N elements", "the second largest", "which is lower from A and B", etc.). An example of a question asking for direct value is "how much were the sales by

brand during last quarter?". An example of question asking for value derived is "which were the 3 brands with highest sales during last quarter?".

In summary, we can state that the questions that belong to the domain of this work are only the questions that ask about **direct simple historic values**. For better understanding, real examples of the questions inside the scope of our domain can be observed in Figure 4.2.

	Examples of questions about direct simple historic values
1	cuánto havn vendido centro y norte esta semana
2	cuánto ingresamos por agua el último trimestre
3	dame las ventas de diciembre de los ultimos 4 años
4	cuánto vendí el mes pasado y este mismo mes el año pasado
5	cuánto han comprado mensualmente de jumex los clientes durante los últimos 12 meses
6	¿cuál es el comportamiento de ventas de agua embotellada durante los fines de semana a partir del primero de enero?
7	para cada almacèn, dame las ventas de jumex, gatorade y refrescos el mes pasado
8	cuánto se vendió de epura de martes a martes
9	¿Cuánto se vendió de refresco en el primer trimestre?
10	ventas los últimos 30 de abril, 10 de mayo, 24 y 31 de diciembre

Figure 4.2: *Examples of questions inside the scope of the domain, taken from the source dataset in Spanish used for this research work.*[4]

3.4. Need for Time Normalization

As observed in figure 4.1, time expressions can affect the request section, the constraints section and the aggregation level section of the SQL query structure. However, it is important to note that this work will focus on the temporal information affecting the constraints section of the query structure, because this section is the most complex. Hence, the translation to temporary sequences is only be based in the temporal information which constraints the query space.

Time Normalization is relevant because it provides a mechanism to translate diverse and ambiguous temporal expressions into exact date values (or period names) that can be found in the source tables for the SQL queries. Frequently, questions contain temporal information that is not expressed in absolute terms; but in relative terms to the actual date and time the questions are asked. For example, to translate the expression found in question of figure 4.1: "*during the first quarter of the year",* first we need to stablish the year as the current year, the corresponding year to the date in which the question is being processed, then we need to address the first quarter inside the current year, and then we have to identify the starting and ending point that delimit the dates enclosed by the first quarter in order to generate precise constraints that can be applied in the source table, like "date>=01/01/2021" and "date<01/04/2021". Time normalization makes this

[4] Chapter 5 introduces the source dataset

possible by defining a mechanism to correctly identify the absolute values that formalize the time constraints, considering the relation distinct time expressions keep to the actual time in which the question is asked. Without time normalization, it would be impossible to perform a query corresponding to a question that contains relative time information.

It is important to note that time normalization performed during this work is limited to handling dates, without including time (hours, minutes, seconds, etc.) information. Thus, dates are the minimum temporal units in the timeline used in this work.

Chapter 4

4. SOURCE DATASET

4.1. Data Collection

Integrating a dataset of natural language questions containing temporal information and asking about precise data contained in a transactional database in the Business domain in Spanish Language, was the necessary first step to identify the Temporal Expressions that need to be integrated in the Ontology proposed by this work, as well as the first step for developing and validating the Set of rules to annotate such questions as temporal normalized sequences containing all temporal constraints necessary to perform a correct search. This dataset was also first step for defining a convenient method for data augmentation and for training, validating and testing the sequence-to-sequence model for translating questions containing temporal expressions into temporal normalized sequences.

As first action for integrating this dataset, it was built a transactional data table with 163,497 records containing information about sales data for a company selling beverages. The table contains the fields “TransactionID” as unique identifier of the sale transaction, the “Date” in which the sale was performed, the “Store” where the sale was performed, the “Category” of the sold beverages, the “Brand” of the beverages and the “Sales” amount in money corresponding to the transaction. A sample of this table can be observed in Figure 4.1 below.

TransactionID	DATE	STOR	CATEGORY	BRAND	SALES
100669442173542869	18-jun-18	SUR	Refresco	Pepsi	$ 85.00
100768642013657417	09-ene-18	SUR	Refresco	Squirt	$ 63.00
100767442178336999	23-jun-18	SUR	Refresco	Pepsi	$ 69.00
100741742362403106	24-dic-18	ESTE	Refresco	Manzanita Sol	$ 168.00
100767442439584884	10-mar-19	SUR	Refresco	Pepsi	$ 133.00
100738842095742136	01-abr-18	OESTE	No Carbonatados	Gatorade	$ 154.00
100226142431791040	02-mar-19	CENTRO	No Carbonatados	Gatorade	$ 86.75
100632542452417289	23-mar-19	ESTE	Refresco	Pepsi	$ 107.00
100344042061746640	26-feb-18	CENTRO	Refresco	7 Up	$ 52.00
100935442111923736	17-abr-18	SUR	Agua Embotellada	Epura	$ 63.00

Figure 4.1: *Example of transactional data table containing information about sales data for company selling beverages.*

As second action for integrating this dataset, people in three distinct groups were requested to generate questions with business sense about the information contained in the transactional data

table, in order to contribute to the development of a virtual agent specialized in responding questions of Business domain[5]. For improving the variability and quality of the questions, people were requested to follow these guidelines:

- **Language:** Do not ask like a computer or an expert data analyst would do it, ask like ordinary people do. It is even ok if you have misspellings.

- **Business Sense:** Even if you think quickly about your questions, please think about questions of interest that a business owner or analyst would ask with the information provided. Asking incoherent questions does not help to train the system and can hurt its subsequent performance.

- **Use Synonyms:** Do not restrict yourself to just using the names of the columns in the table. Take the opportunity to use words that you would commonly use to refer to the same thing. For example: sales, income, profits.

- **Use distinct Expressions:** Don not just think of using different words. We commonly use various expressions to say the same thing, for example: "how much were the sales in", "how much did we sell in", "give me the sales of".

- **Do not to worry about details:** There is no problem if you are missing a detail. Very often, we ask incomplete questions. It is also important to have a sample of incomplete questions, so the system learns to deal with them. For example: "give me the sales of bottled water" (incomplete), "how much we sold per month in water" (incomplete). In contrast, "I want to know the monthly income of the water business for the current year" (complete).

- **Diversity:** Try each new question to be very different from previous ones, without affecting the spontaneity of language, in order to generate many different cases and better train the system.

Ten people from three distinct groups (5 co-workers, 2 relatives and 3 friends) contributed to collect 651 questions. The questions generated by co-workers are used for development, while the questions generated by relatives are used for validation and the questions generated by friends are reserved for final testing.

[5] Original request can be consulted at Annex 1: “Instrument for Question Collection.”

4.2. Data Description

The 720 questions were reviewed to identify the questions that could be responded by the basic SQL query structure this work is focusing on. After this filtering, only 395 questions were found to ask about *direct simple historic values*, which are the questions the basic SQL structure can responds. However, only 265 questions contain temporal information constraining the queries. Hence, the original dataset for the realization of this work is composed by 265 questions that meets the characteristics of the domain explained in previous chapter. From these questions, 73 questions were generated by co-workers and were used for identifying possible elements to include in the set of rules for annotating the questions as temporal normalized sequences, as well as for training the sequence-to-sequence translation model, 96 questions were provided by relatives and are reserved for validation of the seq-to-seq model, meanwhile 96 questions were generated by friends and are kept for testing the seq-to-seq model.

Chapter 5

5. TIME NORMALIZATION

5.1. Understanding of Temporal Expressions

For starting the process of time normalization, the first step is to characterize the temporal expressions we can find in the questions of the domain of interest. This is a description of such temporal expressions:

- **Time Points:** The timeline for this domain is composed by an infinite sequence of finite points, represented by **dates**, the smallest time unit possible. Dates are absolute time points because they indicate precise points in timeline, without depending on any other time point.

- **Time Intervals:** An interval denotes a set of time points between a starting point and an ending point, both limits included or not. For some intervals, the starting and ending point the same time point. Intervals are *"absolute intervals"* when their starting and ending points are defined by **dates** or **time adverbs** (demonstrative adverbs denoting a single time point interval which is relative to the actual date: "today", "yesterday" and "the day before yesterday"), or when they are defined by specific time entities having a well-known starting point and ending point, as it is the case of **years** (e.g., "2018", "2019") or the official duration of an historical event (e.g., "World War II"). Intervals are *"repeating intervals"*[6], when the starting and ending points are not unique points in timeline (they are not dates); but they delimit an interval that appears infinite times in timeline as it is the case of **days of the week** ("Monday", "Tuesday",…,"Friday"), **days of the month** ("1", "2", "3",…,"31"), **months** ("January", "February",…, "December"), **seasons** ("Spring", "Summer", "Autumn", "Winter") and **periodic events** that are common knowledge for everyone (Christmas, Easter Week, Black Friday, Vacations) or expressions (e.g., "January to March", "March 5th to 15th") without complete information for denoting unique points in the timeline.

- **Periods:** Specific amount of time (quantity of time points) expressed by standard units of time (day, week, fortnight, month, bimester, quarter, third, year). As time units, periods allow to express the length of time intervals.

[6] Concept of absolute and repeating intervals has been taken from SCATE (Bethard and Parker, 2016).

- **Temporal Operators[7]:** Adjectives and prepositions that operate over time points, time intervals or periods to generate new intervals or periods ("**From", "To", "After", "Before", "Every", "Minus", "Plus", "Current", "With Respect To", "Last", "Next", "Bottom", "Top", "To Date", "Versus"**). Temporal operators are described in following section.

5.2. TimeSeq: Rules for Annotating Normalized Time Sequences

As explained in Chapter 1, this work proposes to normalize the temporal information contained in questions as temporal normalized sequences, in order to train a model to perform such translation. The following system of rules, named as TimeSeq system, explain how to annotate the temporal information found in questions, considering the characteristics of the time expressions introduced in previous section.

Time intervals

1. **Dates:** Time intervals defined by single dates are annotated as a composition of Year, Month and Day of the Month, respecting such order. Following rules will explain the annotation of years, months and days; but this an example of how an annotated date looks: "2019 01_ENERO 1".
2. **Time Adverbs:** Time intervals defined by time adverbs are annotated with its corresponding term:
   - Today: TODAY
   - Yesterday: YESTERDAY
   - Day before yesterday: BEFORE_YESTERDAY
3. **Years:** Time intervals defined by years are annotated simply with the number corresponding to the year ("2018", "2019").
4. **Months:** Time intervals defined by months are annotated with the Spanish name of the month[8]:
   - January: 01_ENERO
   - February: 02_FEBRERO
   - March: 03_MARZO

[7] Temporal operators were defined considering the questions observed in training set, the operators proposed by SCATE and own experience in current domain.

[8] Cardinal number at the beginning of the month name is added for facilitating sorting during posterior queries.

- April: 04_ABRIL
- May: 05_MAYO
- June: 06_JUNIO
- July: 07_JULIO
- August: 08_AGOSTO
- September: 09_SEPTIEMBRE
- October: 10_OCTUBRE
- November: 11_NOVIEMBRE
- December: 12_DICIEMBRE

5. **Days of a Month:** Time intervals defined by the day of a month are annotated simply with the corresponding cardinal number ("1", "2", "3",…,"31").

6. **Weekdays:** Time intervals defined by days of the week are annotated with the Spanish name of the weekday, including the terms weekend, weekends and working days in English:
   - Monday: 01_LUNES
   - Tuesday: 02_MARTES
   - Wednesday: 03_MIERCOLES
   - Thursday: 04_JUEVES
   - Friday: 05_VIERNES
   - Saturday: 06_SABADO
   - Sunday: 07_DOMINGO
   - Weekend: WEEKEND
   - Weekends: WEEKENDS
   - Working days: WORKINGDAYS

7. **Seasons:** Time intervals defined by seasons are annotated with the Spanish name of the season:
   - Spring: 01_PRIMAVERA
   - Summer: 02_VERANO
   - Autumn: 03_OTOÑO
   - Winter: 04_INVIERNO

8. **Events:** Time intervals defined by events are annotated with the name of the event. These are the events considered in this work:
   - Black Friday: EVENT:BUEN_FIN
   - Easter Week: EVENT:SEMANA_SANTA
   - Christmas: EVENT:NAVIDAD
   - Vacations: VACATIONS[9]

[9] Vacations intersect with Seasons to form a new interval. For example, "Summer Vacations".

Time Interval Composition

9. **Periods:** Periods are annotated with their English names. Since business domain queries focuses in recent years, the year is the maximum period contained in these annotation rules. All periods are annotated with corresponding number[10] (singular or plural) when required, a aligned to original question. Time adverbs “today”, “yesterday” and “before_yesterday” are used for referring to the singular form of “day” period in diverse cases:

    - DAY, DAYS
    - FORTNIGHT, FORTNIGHTS
    - WEEK, WEEKS
    - MONTH, MONTHS
    - BIMESTER, BIMESTERS
    - QUARTER, QUARTERS
    - THIRD, THIRDS
    - SEMESTER, SEMESTERS
    - YEAR, YEARS

10. **Cardinal numbers:** Besides their use for representing specific days of the month, use cardinal numbers for representing a certain quantity of periods. Cardinal numbers are annotated directly with their corresponding digits. Cardinal numbers are always followed by a period with plural number; except when they are representing days of the month, in which case they are annotated alone.

11. **Ordinal numbers:** Use to specify the interval constituted to the N-th period from the total periods in which an interval can be divided, like in the expression “day 4th of the month”. Ordinal numbers are always followed by a period name with single number. Ordinal numbers are annotated considering the gender[11] of the following period name in Spanish Language. Thus, ordinal numbers are annotated as follows:

    - 01o, 01a
    - 02o, 02a
    - 03o, 03a
    - 04o, 04a
    - 05o, 05a
    - 06o, 06a
    - 07o, 07a
    - 08o, 08a
    - 09o, 09a
    - 10o, 10a
    - 11o, 11a
    - 12o, 12a

[10] This is necessary to perform the anonymization process of data augmentation method, explained in Chapter 6.

[11] This is necessary to perform the anonymization process of data augmentation method, explained in Chapter 6.

12. **From Operator (FROM):** Use for indicating the starting point of a time interval, including the starting point. Always, annotate “FROM” followed by an interval. Examples: “from April 1st, 2019” = “FROM 2019 04_Abril 1”, “from June” = “FROM JUNE”.

13. **To Operator (TO):** Use for indicating the ending point of a time interval, including the ending point. Always, annotate “TO” followed by an or interval. Examples: “to May 1st, 2019”= “TO 2019 01_MAYO 1”, “to August” = “TO 08_AGOSTO”. When having an interval spanning from a certain repeating interval to the next occurrence of the same repeating interval, it will be understood that interval annotated before the “TO” operator is the interval occurred just before the interval after the operator. Example: “from Tuesday to Tuesday” “FROM 02_MARTES TO 02_MARTES”.

14. **After Operator (AFTER):** Use for indicating the starting point of a time interval, without including the starting point. Always, annotate “AFTER” followed by an interval. Sometimes, this operator can be preceded by periods or repeating intervals in order to form a whole new interval integrated by the time information to the left an to the right of the “AFTER” operator. Examples: “after June” = “AFTER JUNE”, “after May 31th and before July 1st ” = “AFTER 05_Mayo 31 BEFORE 07_JULIO 1”.

15. **Before Operator (BEFORE):** Use for indicating the ending point of a time interval, without including the ending point. Always, annotate “BEFORE” followed by an interval. Examples: “before August” = “BEFORE 08_AGOSTO”, “before May 1st, 2019” = “BEFORE 2019 01_MAYO 1”.

16. **Every Operator (EVERY):** Use to refer to all repeating intervals of the same sequence, which are contained in available data or in a certain interval of reference, preceding the every operator. This operator also works with years in the same way that it works with repeating intervals. Write “EVERY” followed by period singular name. Examples: “for all years” = “EVERY YEAR”, “every weekend of last summer” = “LAST 02_VERANO EVERY WEEKEND”.

17. **Ago Operator (AGO):** Use to move an interval N periods earlier in timeline, i.e., it defines a new interval, whose starting point is located N periods before the starting point of a reference interval, while keeping the same length of such interval. When N = 1, write the reference interval followed by “AGO”, which must be followed by period singular name.

When N >1, write the interval followed by “AGO”, which must be followed by cardinal number N and then, by the period plural name. When the interval of reference is the actual date, use the interval “TODAY”. Examples: “a month ago” = “TODAY AGO MONTH”, “2 last quarters and same period 2 years ago” = “LAST 2 QUARTERS AGO 2 YEARS”.

18. **Since Operator (SINCE):** Use this together with “Ago Operator”, to include N intervals with length equivalent to a reference interval that exist at equal distance between the new starting point denoted by “Ago Operator” and the starting point of the reference interval. Always write the interval followed by “SINCE AGO”, which must be followed by cardinal number N and then, by the period plural name. Example: “2 last quarters and same period in the last 2 years” = “LAST 2 QUARTERS SINCE AGO 2 YEARS”, “summer to date and same period during last 2 years” = “LAST 02_VERANO TODATE SINCE AGO 2 YEARS”.

19. **Minus Operator (MINUS):** Use to indicate a new interval with length equivalent to N rolling periods and starting point located N rolling periods before a certain interval of reference, where a rolling period is an artificial period created with the standard length in days of an indicated natural period (e.g., month=30 days, week=7 days). When N = 1, write the interval of reference followed by “**MINUS**” and by period singular name. When N >1, write the reference interval followed by “**MINUS**” and by cardinal number N and by the plural name of period. Examples: “counting 1 year before date” = “TODAY MINUS YEAR”, “3 rolling weeks before May 10th” = “LAST 05_MAY0 10 MINUS 3 WEEKS”.

20. **Plus Operator (PLUS):** Use to indicate a new interval with length equivalent to N rolling periods and ending point located N rolling periods after a certain interval of reference, where a rolling period is an artificial period created with the standard length in days of an indicated natural period (e.g., month=30 days, week=7 days). When N = 1, write the interval of reference followed by “**PLUS**” and by period singular name. When N >1, write the reference interval followed by “**PLUS**” and by cardinal number N and by the plural name of period. Examples: “1 week after of July 1st” = “LAST 07_JULIO 1 PLUS WEEK”, “3 months counted after May 10th” = “LAST 05_MAY0 10 PLUS 3 MONTHS”.

21. **Current Operator (CURRENT):** Use to indicate the absolute interval including the actual date, which has the length of the referred period. Write “CURRENT” followed by period singular name. Example: “this month” = “CURRENT MONTH”.

22. **Previous Operator (PREVIOUS):** Use to indicate a set of intervals constituted by the N nearest repeating intervals of the same kind, occurred immediately before the starting point of a certain interval of reference. This operator also works with years in the same way that it works with repeating intervals. When N = 1, write "**PREVIOUS**" followed by period singular name or interval. When N >1, write first "**PREVIOUS**", followed by cardinal number N and, then, by the plural name of interval or period. When the interval of reference is the actual date, this is not annotated. When annotating an interval of reference distinct to the actual date, write the interval of reference followed by "PREVIOUS". When referring to a previous interval equivalent to the interval of reference, it is not necessary to annotate the interval after "PREVIOUS". Examples: "previous 4 weekends (from today)" = "PREVIOUS 4 WEEKENDS", "2 months before vacations" = "VACATIONS PREVIOUS 2 MONTHS", "2018 Christmas and 2 weeks earlier" = "2018 EVENT:NAVIDAD VS PREVIOUS 2 WEEKS", "this January compared to previous one"= "01_ENERO VS PREVIOUS".

23. **Next Operator (NEXT):** Use to indicate a set of intervals constituted by the N nearest repeating intervals of the same kind, occurred immediately after the ending point of a certain interval of reference. This operator also works with years in the same way that it works with repeating intervals. When N = 1, write "**NEXT**" followed by period singular name or interval. When N >1, write first "**NEXT**", followed by cardinal number N and, then, by the plural name of interval or period. When the interval of reference is the actual date, this is not annotated. When annotating an interval of reference distinct to the actual date, write the operator "WRT" at the end of previous expression, followed by the interval of reference. Examples: "next 4 weekends (from today)" = "NEXT 4 WEEKENDS", "2018 and next year" = "2018 VS NEXT YEAR WRT 2018", "2018 and next 2 weeks" = "2018 VS NEXT 2 WEEKS WRT 2018".

24. **Last Operator (LAST):** This operator has two use cases. In the first case, it is used to indicate a set of intervals constituted by the N nearest repeating intervals of the same kind, occurred immediately before current date. This operator also works with years in the same way that it works with repeating intervals. When N = 1, write "LAST" followed by period singular name or interval. When N >1, write first "LAST", followed by cardinal number N and, then, by the plural name of interval or period. Examples: "last 4 weekends (from today)" = "LAST 4 WEEKENDS", "last Friday" = "LAST 05_VIERNES". In the second case, this operator is used to indicate a sub-set of time points taken from a time interval, where this sub-set is constituted by the time points included in the last N periods from the total periods in which the original interval can be divided. In this case, the "LAST" operator is **always**

preceded by an interval. Thus, the ending point corresponds to the ending point of the original interval, while the starting point is located N periods earlier. When N = 1, write "**LAST**" followed by period singular name. When N >1, write first "**LAST**", followed by cardinal number N and, then, by the plural name of period. Examples: "last 2 weeks of each Summer" = "EVERY 02_VERANO LAST 2 WEEKS", "last 3 days of March" = "LAST 03_MARZO LAST 3 DAYS" ,"last day of last month" = "LAST MONTH LAST DAY". LAST operator is the only operator than can be omitted in annotation, its absence implies its use. For example: "sales of Tuesday (last Tuesday without explicit mention)" = "TUESDAY".

25. **Top Operator (TOP):** Use to indicate a sub-set of time points taken from a time interval, where this sub-set is constituted by the time points included in the first N periods from the total periods in which the original interval can be divided. Thus, the starting point corresponds to the starting point of the original interval, while the ending point is located N periods later. When N = 1, write "**TOP**" followed by period singular name. When N >1, write first "**TOP**", followed by cardinal number N and, then, by the plural name of period. Examples: "first 3 days of March" = "LAST 03_MARZO TOP 3 DAYS" ,"first day of last month" = "LAST MONTH TOP DAY", "first 3 months of the year" = "CURRENT YEAR TOP 3 MONTHS".

26. **To Date Operator (TODATE):** Use to indicate a new interval with homologous time span to the one in the interval "current period to date"; but located at the beginning of other intervals defined by the same kind of period. This operator, firstly, identifies the Delta (number of time points) occurred between the actual date (included) and the starting point (not included) of the current interval that includes the actual date and has the length of a referred interval. Then, it generates a new interval whose starting point corresponds to the starting point of such referred interval and has a length of Delta. Write "TODATE" only after the referred period with singular name. Example: "year to date and same period last year" = "CURRENT YEAR VS LAST YEAR TODATE".[12]

Sequences of Intervals

27. **Sequence Composition**: Temporal sequences are sequences of time intervals. Numbers, periods and operators can only appear in a sequence for describing intervals, they lack of

[12] This example shows that it is not necessary to annotate the operator TODATE for current period, since this is implicit.

meaning by themselves and, only, when composing an interval, they have a meaning in the sequence.

28. **Sequence Logics:** Sequences reads left to right. Intervals must appear in descending order according to their time length. An interval B appearing to the right of interval A indicates the intersection of the sets of time points corresponding to intervals A and B. An interval C appearing to the right of the previous interval B indicates the intersection of the sets of time points corresponding to intervals A B C. Examples: “2019 01o QUARTER”, “2018 3o MONTH 1o WEEK”. Figure 6.1 illustrates how intervals integrate to compose a time sequence, reading from right to left:

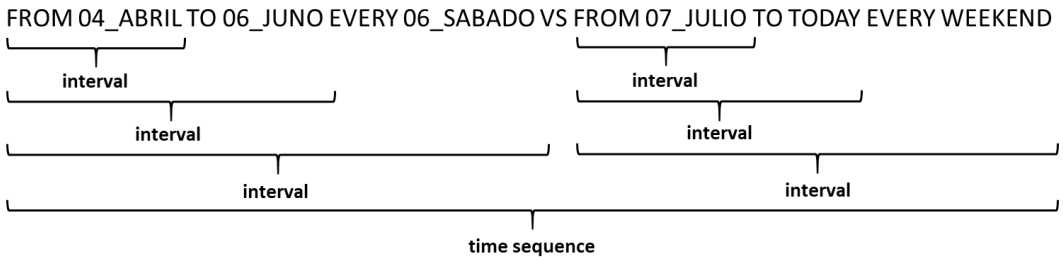

Figure 6.1: *Example of temporal sequence as composition of time intervals, for the phrase: “every Saturday from April to June and every weekend from July to date”.*

29. **Versus Operator (VS):** Intervals that are not intended to intersect must be separated by using the “versus” operator. Example: “February of this year and March of previous year” = “CURRENT YEAR 02_FEBRERO VS LAST YEAR 03_MARZO”.

30. **Constraint 1**: two or more intervals with the same length that do not intersect; but are intended to intersect with common intervals of distinct length must be annotated in same sequence, instead of distinct sequences. Such intervals with same length must be sorted intervals in chronological order: Example: “Last week of January and February of 2019” = ”2019 01_ENERO 02_FEBRERO LAST WEEK”.

31. **Constraint 2**: In case the reference interval for an operator “Ago”, “Since”, “Plus”, “Minus”, “Previous” or “Next” is followed by an operator “Vs”, “To” or “After”, the reference interval must be omitted before the operator “Ago”, “Since”, “Plus”, “Minus”, “Previous” or “Next”, considering the relation is implicit in the sequence. Example: “January vs pevious year” =

"01_ENERO VS PREVIOUS YEAR", "from December 12th, 2018 to February 2nd of next year" = "FROM 2018 12_DICIEMBRE 12 TO NEXT YEAR 02_FEBRERO 2".

5.3. Additions to training set for representing temporal expression not contained in original data

Original dataset was adjusted to include new sequences containing temporal classes that were not present in original sequences or that had few examples. Before executing the augmentation algorithm, new sequences were added to assure the dataset had at least 5 examples of each ontology class for training. This way, the original 265 questions with temporal information constraining the queries were complemented with 42 new questions, for reaching a total of 307 questions (115 for training, 96 for validation and 96 for testing).

Chapter 6

6. DATA AUGMENTATION

6.1. Overview

The data augmentation process works in 2 stages: (1) Anonymization of Paraphrases, and (2) Combination of Paraphrases, which are described in following sections. Applying these stages, the data augmentation, creates new pairs of questions and time sequences without affecting the reliability of the temporal information, as result of controlled combinations of paraphrases substituted at original pairs and at the new pairs generated. Augmentation is performed only for learning and validation dataset, since test data is reserved for testing without being affected by any possible bias derived from augmentation method. Because of time limits, we restricted the generation of new questions to 20 thousand new questions for each original question. This way, the initial 115 pairs of sequences in the training data are exploded to 605,158, while the initial 96 pairs for validation are exploded to 498,737. As can be inferred, there are various pairs that generate less than 20 thousand new questions.

6.2. Ontologies of Paraphrases

These ontologies are used by the Data Augmentation Algorithm for identifying the paraphrases that can be replaced in every question, and later, they are used to identify the specific values that can replace them. Because of the particularity of the domains covered in this work and the fact that the questions analyzed are in Spanish language, we decided it was relevant for the success of this project to develop our own ontology of paraphrases. This way, we integrated 3 ontologies:

(1) **Ontology of Paraphrases for Temporal Expressions:** containing all paraphrases expressing temporal information. This ontology can be used for any domain.

(2) **Ontology of Paraphrases for General Business Database Querying:** integrating all paraphrases related to common expressions used when asking for information contained in a business database. This ontology can be used for any other business domain.

(3) **Ontology of Paraphrases for Specific Industry Domain:** collecting all paraphrases related to the specific domain in which all questions are formulated in this project. This is the only ontology that needs to be replaced when applying augmentation method in other industry.

All ontologies are composed by 3 hierarchical levels: (1) supra-categories, (2) categories and (3) paraphrases, as observed in figure 6.1.

Supra-Category	Category	Paraphrase
ADVERB_MONTH_PERIOD	EVERY DAY	'a cierre de cada dia'
		'a fin de cada dia'
		'a final de cada dia'
		'a termino de cada dia'
		'de manera diaria'
		'en forma diaria'
		'de forma diaria'
		'diario'
		'diaria'
		'diariamente'
		'por dia'
		'por dias',
		'en cada dia'
		'para cada dia'
		'de cada dia'
		'por cada dia'
		'cada dia'
		'diarias'
		'diarios'
	EVERY WEEK	
	EVERY FORTNIGHT	

Figure 6.1: *Example of the supra-category "ADVERB_MONTH_PERIOD" in the Ontology of Temporal Expressions.*

6.3. Algorithm for Augmentation

Anonymization of Paraphrases

1. For the question of each pair of sequences:

    1.1. For each category of the 3 ontologies:

        1.1.1. For each paraphrase in the category:

            1.1.1.1. If the paraphrase exists in the question, replace it by the corresponding category, and continue evaluating the question with next paraphrase.

Combination of Paraphrases

1. Create an empty list named “list0” for containing total new pairs of questions and time sequences to be generated.
2. For each pair of anonymized question and time sequence:
    2.1. Create a list named “list1” for containing new pairs of questions and time sequences that will be generated from current pair. Append the current pair to list1.
    2.2. Check in the question has any anonymized paraphrase:
        2.2.1. If the question has not anonymized paraphrase:
            2.2.1.1. Append pair to list0.
            2.2.1.2. Continue with next pair.
        2.2.2. If the question has an anonymized paraphrase:
            2.2.2.1. For each category in the complete ontology:
                2.2.2.1.1. Check if the category is contained in the anonymized question:
                    2.2.2.1.1.1.If the category is not contained:
                        2.2.2.1.1.1.1. Continue with next category.
                    2.2.2.1.1.2.If the category is contained:
                        2.2.2.1.1.2.1. Create an empty list named “list2” for containing new pairs of question and time sequence will be generated from current pair and ontology category.
                        2.2.2.1.1.2.2. For each pair contained currently in list1:
                        2.2.2.1.1.2.3. if the category corresponds to the Ontology of Paraphrases for Specific Industry Domain, integrate a list including a sample of all the equivalent paraphrases corresponding to the supra-category (paraphrases of distinct categories which can replace the original paraphrase, without affecting the relevant information to be learned by seq-2-seq model).
                            2.2.2.1.1.2.3.1. From the original question, generate a new question replacing the category by the first element in the list with the sample of equivalent paraphrases. From the original question, continue generating new questions until finishing the equivalent paraphrases. Append each pair of new question with its corresponding time sequence without change, into list2.

2.2.2.1.1.2.4. if the category corresponds to the Ontology of Paraphrases for Temporal Expressions, integrate a list including a sample of all the equivalent paraphrases corresponding to the supra-category.

2.2.2.1.1.2.4.1. Only in the case the category is present in the question and in the time sequence, use original question and sequence to generate a new question and a new time sequence, replacing the category by the first element in the list with the sample of equivalent paraphrases. Continue generating new questions and corresponding time sequence until finishing the equivalent paraphrases. Append each new pair of question and time sequence into a list2.

2.2.2.1.1.2.5. If the category corresponds to the Ontology of Paraphrases for General Business Database Querying, integrate a list including all the equivalent paraphrases corresponding to the category.

2.2.2.1.1.2.5.1. From the original question, generate a new question replacing the category by the first element in the list with the sample of equivalent paraphrases. From the original question, continue generating new questions until finishing the equivalent paraphrases. Append each pair of new question with its corresponding time sequence without change, into a list2.

2.2.2.1.1.2.6. Copy pairs of list2 into list1.

2.3. Append each new pair of question and time sequence into a list0.

3. Repeat N times the steps 1 and 2, using list0 as new input.

Chapter 7

7. EXPERIMENTS

7.1. Experimentation Outline

This chapter describes the experiments performed to deliver an integral answer to the research objectives mentioned in chapter 1:

- **Research Objective 1.2.** Evaluate if ML/DL models can learn to summarize the temporal information contained in a natural language question into a normalized sequence.
- **Research Objective 2.4.** Evaluate if models trained with this Data Augmentation method can generalize to unseen datasets of questions.

Following sections will present a brief description of each experiment and main results obtained. All experiments involve the training of DL models to learn how to translate questions into time sequences. Model results are obtained for a learning set and a validation set, presenting its accuracy (where accurate sequence translation gets 1 only in the case that true sequence and predicted sequence are equal, token by token) and its capability to generalize in the validation set, as observed by the behavior of the loss function.

Experiment 2 establishes a baseline. Experiment 2 intends to respond if augmentation of data using paraphrases has real benefit. Experiments 3, 4 and 5 aim to improve validation accuracy and model generalization. Experiments 1 to 5 are executed with an implementation of a Transformer[13] model publicly available at Keras documentation[14], keeping same parameter configuration for all experiments (batches of size 64, input embeddings of size 256, a 2048 latent dimensionality of the encoding space, 8 attention heads, one dropout layer with 0.5 rate, among others)[15].

[13] Vaswani et al. (2017)

[14] (Chollet, F., 2021)

[15] Experiments were executed also with batch sizes of 16, 32, 256 and 2048 without altering main conclusions.

Experiment 6 looks for validating previous results obtained with a Transformer architecture by comparing them with a Bi-Directional[16] RNN[17] model with attention[18], which keeps batches of size of 64 and input embeddings of 256, while having 128 LSTM[19] units. This model is an adaptation of the sequence-to-sequence model at character level available at Keras documentation[20]. Experiment 7 obtain the results of final model selected, when applied to the test data.

7.2. Experiment 1: Baseline (Training with data not augmented with paraphrases)

This experiment stablishes the starting point for accuracy and model generalization, before using data augmentation with paraphrasis. It uses sampling with replacement for augmenting data.

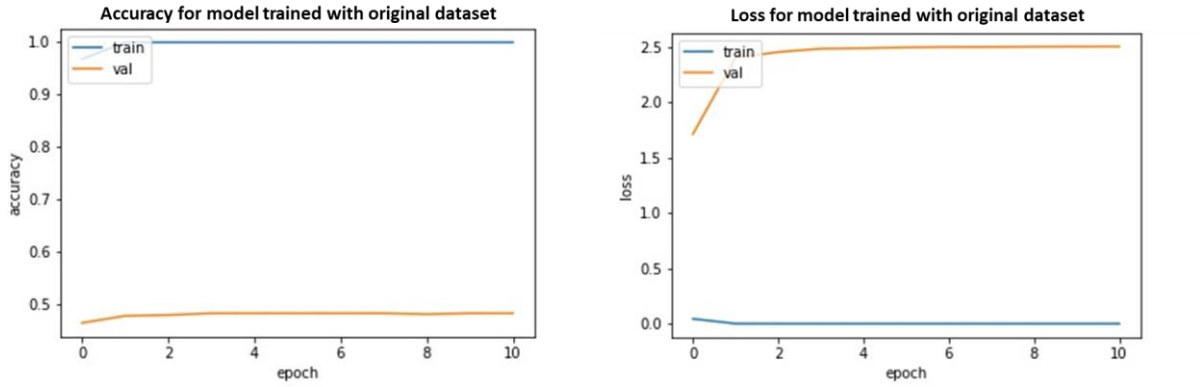

Figure 7.1: *Accuracy and Loss evolution during training of model with original data augmented using sampling with replacement (Training Sample = 200k, Validation Sample = 100k, Validation Accuracy in best epoch = 0.482)*

As observed in figure 7.1, model reaches accuracy of 1 in the training set at the second epoch; while is not capable of continue improving accuracy for validation set beyond 0.482. When observing the loss value, it keeps practically without change after epoch 2 for the training set; while in the validation set, the loss increases quickly from epoch 0 to 3 and it keeps without significant

[16] Schuster & Paliwal (1997)

[17] Cho et al. (2014)

[18] Luong et al. (2015)

[19] Hochreiter & Jürgen Schmidhuber (1997)

[20] (Chollet, F., 2017)

change after epoch 4. In summary, starting point for accuracy in validation set is 0.482 and this model is overfitted and not able to generalize.

7.3. Experiment 2: Evaluate impact of using augmented data with paraphrases

In a first instance, this experiment trains with augmented data using paraphrases, and performs validation with not augmented data. In a second instance, the experiment performs validation with augmented data to identify if there is a possible bias derived from the augmentation method.

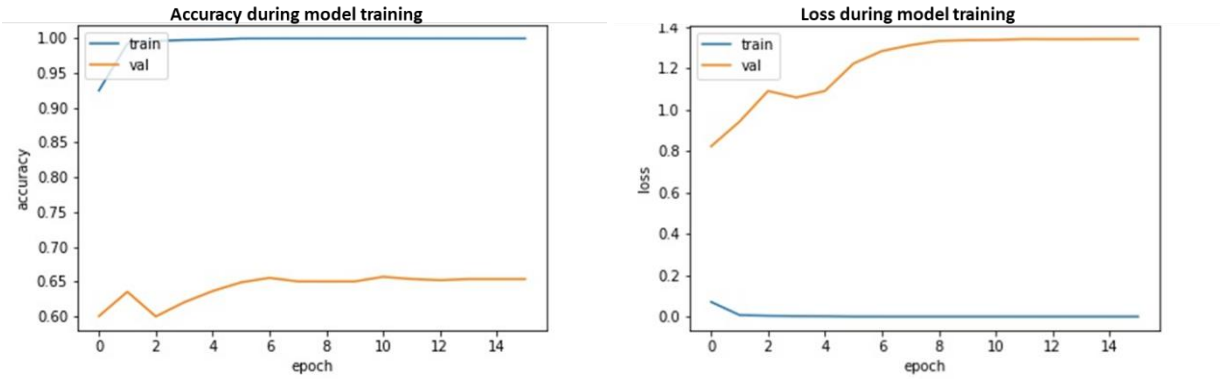

Figure 7.2: *Accuracy and Loss evolution for model trained with augmented data and validated with not augmented data (Training Sample = 200k, Validation Sample = 100k, Validation Accuracy in best epoch = 0.657)*

As observed in figure 7.2, model reaches accuracy of 1 in the training set at the second epoch; while is not capable of improving validation accuracy of 0.65 beyond epoch 6. When observing the loss value, it keeps practically without change after epoch 2 for the training set; while in the validation set, the loss increases continuously from epoch 0 to 8 and the continues without relevant change. Since accuracy and validation shows similar behaviors for all experiments, the description of the curves will be omitted for posterior experiments. In summary, model trained with data augmentation reaches an accuracy 0.175 above the baseline, but this model continues overfitting and is not able to generalize.

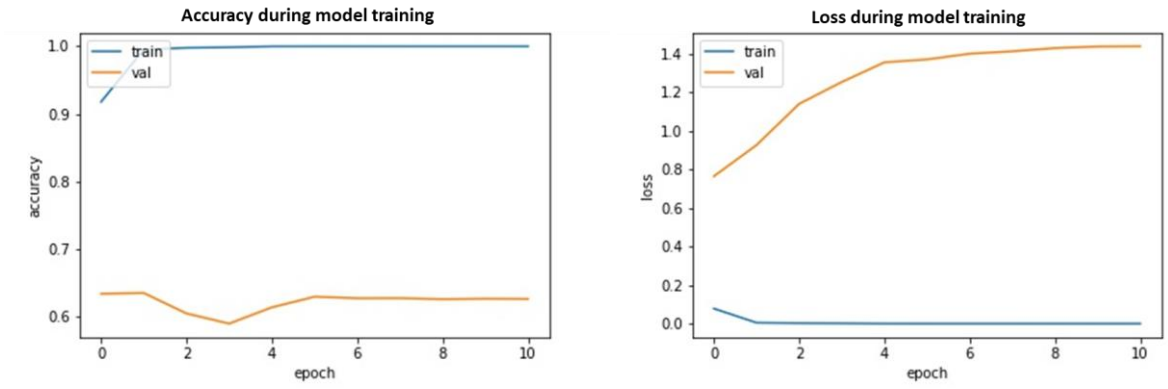

Figure 7.3: *Accuracy and Loss evolution for model trained and validated with augmented data (Training Sample = 200k, Validation Sample = 100k, Validation Accuracy in best epoch = 0.635)*

As observed in figure 7.3, accuracy reaches 0.635 for the validation with augmented data, while validation loss increases continuously from epoch 0 to 10. In summary, this model is also overfitted; but it reaches a validation accuracy 0.019 below the validation accuracy for the data not augmented with paraphrases. This evidences there is not a bias derived from the data augmentation with paraphrases, that could contribute to get better results when comparing augmented training data against augmented validation data.

7.4. Experiment 3: Evaluate impact of training with distinct amount of augmented data

This experiment trains a first model with 20k examples of augmented data, and a second model with 600k examples and compare results with last model of previous experiment, which was trained with 200k of augmented data. Validations are performed against sets of augmented data.

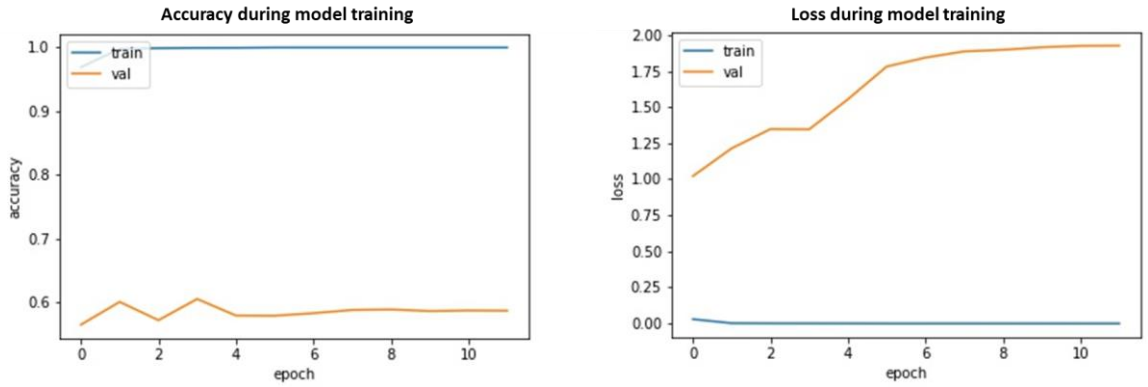

Figure 8.4: *Accuracy and Loss evolution for model trained and validated with augmented data (Training Sample = 600k, Validation Sample = 300k, Validation Accuracy in best epoch = 0.635)*

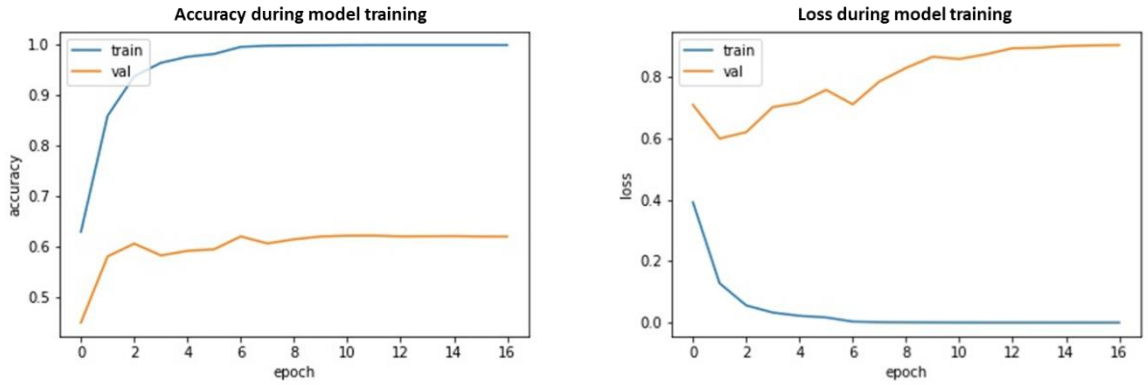

Figure 8.5: *Accuracy and Loss evolution for model trained and validated with augmented data (Training Sample = 20k, Validation Sample = 10k, Validation Accuracy in best epoch = 0.622)*

When comparing figures 7.3, 7.4 and 7.5, it is observed that training with more data than 200k instances do not improve accuracy or avoid overfitting. On the other hand, a model trained with less

data requires more epochs to reach accuracy convergence, and result is slightly lower (only 0.013 below validation accuracy obtained with model trained with 200k instances).

7.5. Experiment 4: Evaluate impact of training with sample balance by original sequence group and by cluster of sequences

The number of synthetic questions generated for each original question varies depending on the number of possible combinations that arise from the composition of each original question. New question generation was limited to a maximum of 20k per original question; but there are several questions that do not reached that limit, some questions where not even able to generate more than a hundred of synthetic questions. For this reason, there is the risk that unbalance of the augmented sample could avoid the model learn from examples with low representation.

To evaluate the impact of unbalanced sample, this experiment trains a first model on a dataset completely balanced by question groups (synthetic questions grouped by the source sequence that was combined to originate such questions). Later, it trains a second model on a dataset balanced among 10 clusters identified with a simple k-means algorithm, using a set of features obtained from the time sequence.

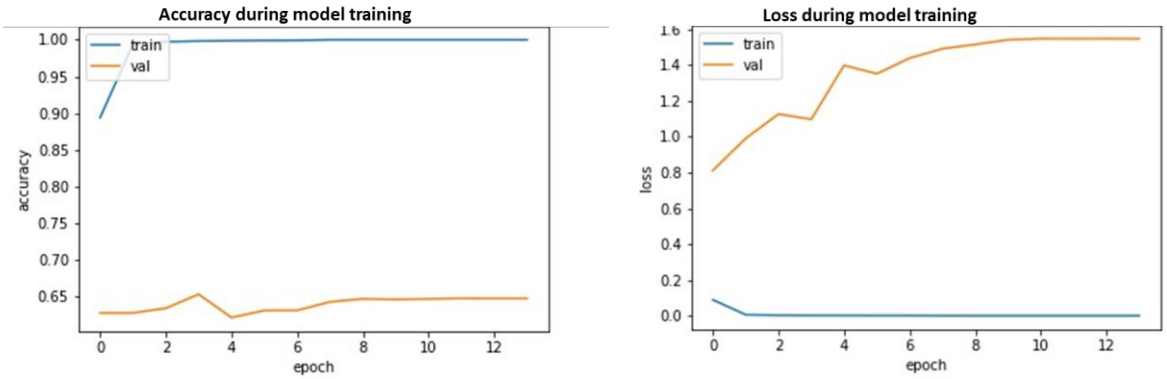

Figure 7.6: *Accuracy and Loss evolution for model trained with sample balanced by original question group (Training Sample = 200k, Validation Sample = 100k, Validation Accuracy in best epoch = 0.653)*

As observed in figure 7.6, the model trained with sample uniformly distributed for every group of original questions is reaching a validation accuracy of 0.65 (0.05 above model trained with

unbalanced data). In summary, balancing the data for assuring each original question is entering to the data with equal probability helps to improve the accuracy; but the model is still not generalizing. Model was also validated against data not augmented with paraphrases, reaching an accuracy of 0.64.

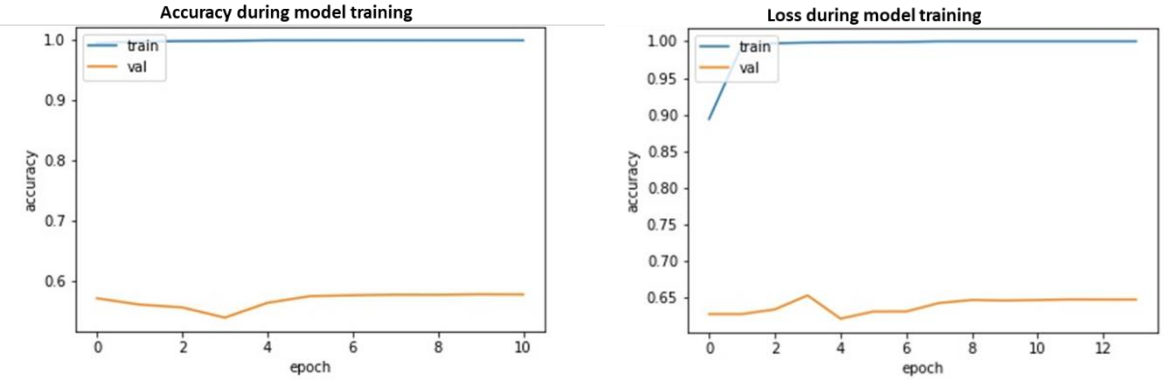

Figure 7.7: *Accuracy and Loss evolution for model trained with sample balanced by cluster (Training Sample = 200k, Validation Sample = 100k, Validation Accuracy in best epoch = 0.577)*

Unexpectedly, figure 7.7 shows the model trained with sample balanced by cluster reaches a validation accuracy of 0.577 (0.023 below model with unbalanced data). This is explained as an effect of poor quality of clusters. For better understanding, models were built for every cluster, finding the validation accuracy (for validation data augmented with paraphrases) improved over the 0.6 validation accuracy of the model with unbalanced data, only for clusters 1 and 6 as presented in figure 8.8 below. For this reason, a “general model” was trained with all clusters, excluding 1 and 6, reaching only a validation accuracy of 0.501 on the data augmented with paraphrases. However, when validating with data not augmented with paraphrases, the general model increased validation accuracy to 0.71 and the model for cluster 6 increased accuracy to 0.875; but the model for cluster 1 decreased validation accuracy to 0.615.

Cluster represented in Validation Data	**Learning data examples** (thousands)	**Learning data examples** (thousands)	**Validation Accuracy** (for validation data augmented with paraphrases)	**Validation Accuracy** (for validation data NOT augmented with)
0	37	13	0.414	*Not calculated*
1	142	77	**0.841**	**0.615**
3	37	13	0.354	*Not calculated*
4	53	25	0.379	*Not calculated*
6	91	50	**0.729**	**0.875**
7	41	23	0.358	*Not calculated*
9	25	20	0.448	*Not calculated*
All, excluding 1 and 6	200	100	**0.501**	**0.710**

Figure 7.8: *Validation Accuracy for models trained by cluster*

In summary, since only cluster 6 improved validation accuracy without exhibiting risk of bias (cluster 1), it was preferred to continue working with only one model for further experiments. This way, the model trained with sample balanced by original question group (validation accuracy = 0.65) is used for further experiments.

7.6. Experiment 5: Evaluate impact of increasing dropout rate

Since previous models are clearly overfitting, new models were trained with higher rates of dropout. Figure 7.9 shows the results for dropout rate of 0.9:

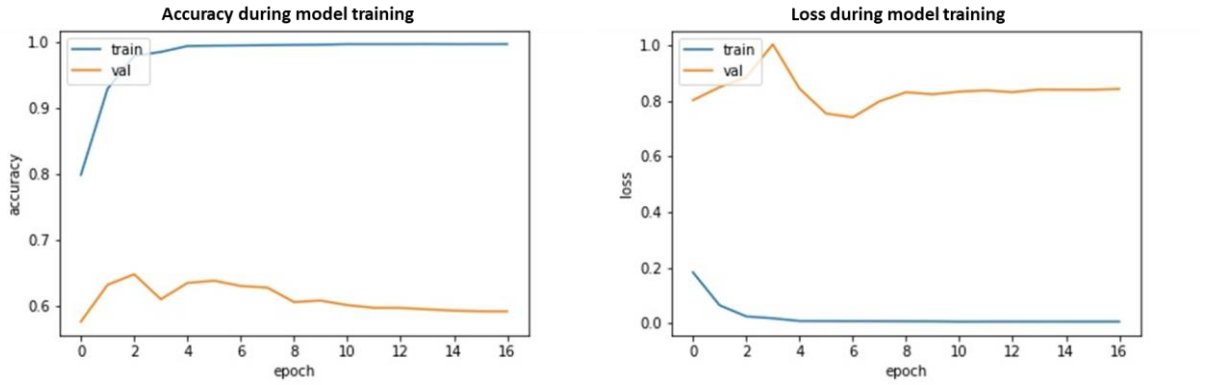

Figure 7.9: *Accuracy and Loss evolution for model trained with sample balanced by original question group, with dropout rate=0.9 (Training Sample = 200k, Validation Sample = 100k, Validation Accuracy in best epoch = 0.638)*

As observed, loss for validation data falls between epochs 2 and 6 and then is continue growing until reaching almost a steady level at last epochs. When observing validation accuracy, this increases during epochs 3, 4, 5 and then starts falling to reaching a steady level around 0.6 at last epochs. Among all executed experiments, this is the only model exhibiting an increasing trend of validation accuracy for a decreasing trend of validation loss, at least, for the discussed epochs. This helps to explain the models are learning very fast because the sample is not providing enough variability. When using a high number of examples to learn at each iteration (dropout rate=0.5), most of the models memorize almost perfectly in 2 epochs; but when reducing the number of examples proportionated in each iteration by setting dropout rate to 0.9, they delay the extend the learning for some additional epochs, letting to observe better the evolution of accuracy and loss in the validation data, before reaching the "steady" levels for accuracy and loss that are observed across all model curves. In summary, models are memorizing the variability provided by training samples; but are not able to generalize to all the variability present at unseen data.

7.7. Experiment 6: Validate previous results with distinct model architecture

This experiment looks for validating the results obtained with Transformer Models, by training a Bi-Directional RNN model with 128 LSTM units and attention, keeping same batch size and input embedding size as in previous models, with 0.5 dropout rate. The model is trained in the sample balanced by original question group an is validated with 3 datasets: (1) validation data augmented with sampling with replacement, (2) validation data augmented with paraphrases without balancing sample and (3) validation data augmented with paraphrases and balancing sample by original question group.

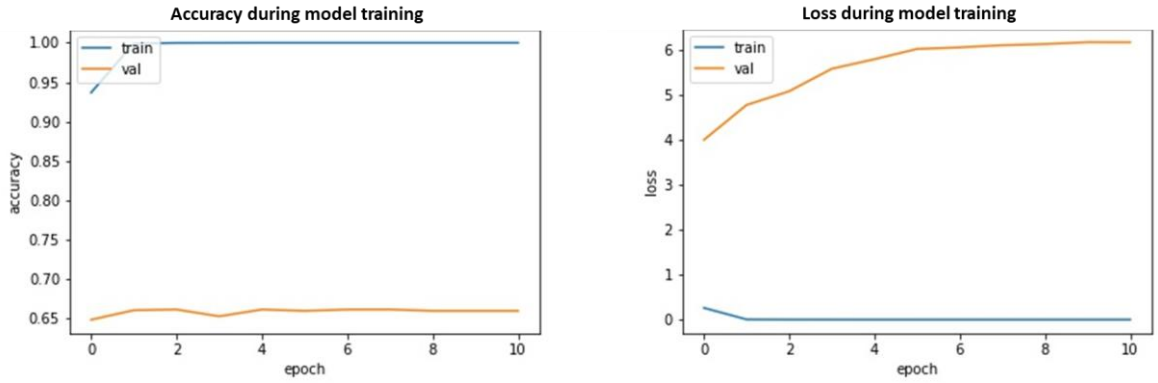

Figure 7.10: *Accuracy and Loss evolution for bi-RNN with attention, trained with sample balanced by original question group and validated with data not augmented with paraphrases, without sample*

balancing (Training Sample = 200k, Validation Sample = 100k, Validation Accuracy in best epoch = 0.661)

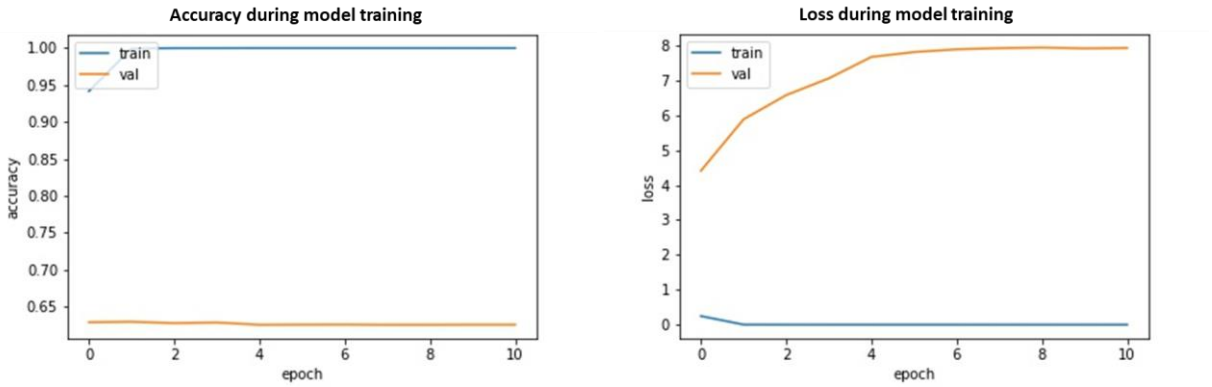

Figure 7.11: *Accuracy and Loss evolution for bi-RNN with attention, trained with sample balanced by original question group and validated with data augmented with paraphrases without sample balancing (Training Sample = 200k, Validation Sample = 100k, Validation Accuracy in best epoch = 0.629)*

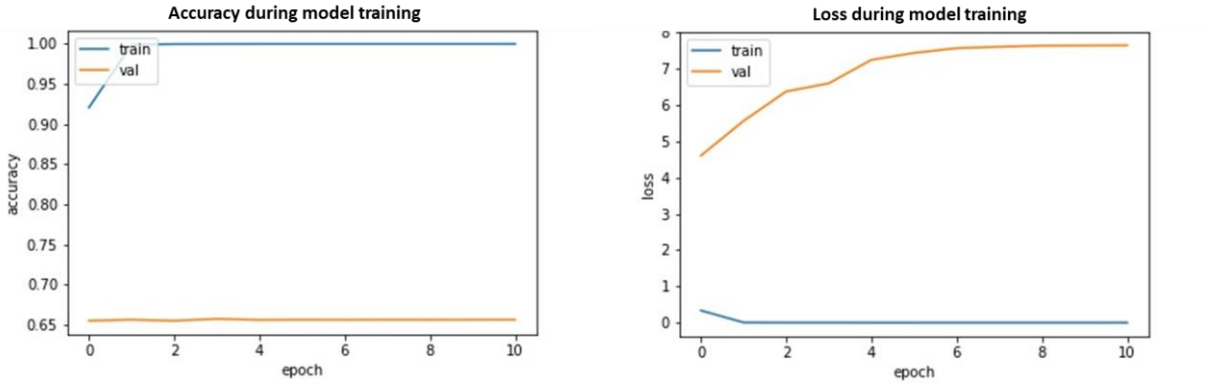

Figure 7.12: *Accuracy and Loss evolution for bi-RNN with attention, trained and validated with sample balanced by original question group (Training Sample = 150k, Validation Sample = 75k, Validation Accuracy in best epoch = 0.657)*

In summary, the results obtained by Bi-RNN model confirm previous results obtained with Transformer models. The model reaches similar accuracy for the three validation sets (0.6229 to 0.657) and is overfitted since the second epoch. This confirms the problem is in the data and not with model implementation.

7.8. Experiment 7: Results for Test Data

Finally, the Transformer model trained with sample balanced by original question group, considered as final model, was evaluated in the test set. The weights of the first epoch were used for testing, reaching an accuracy of **0.698** when directly evaluating the test with 96 questions. Figure 7.13 shows the behavior of testing set in the case it had been used for validation.

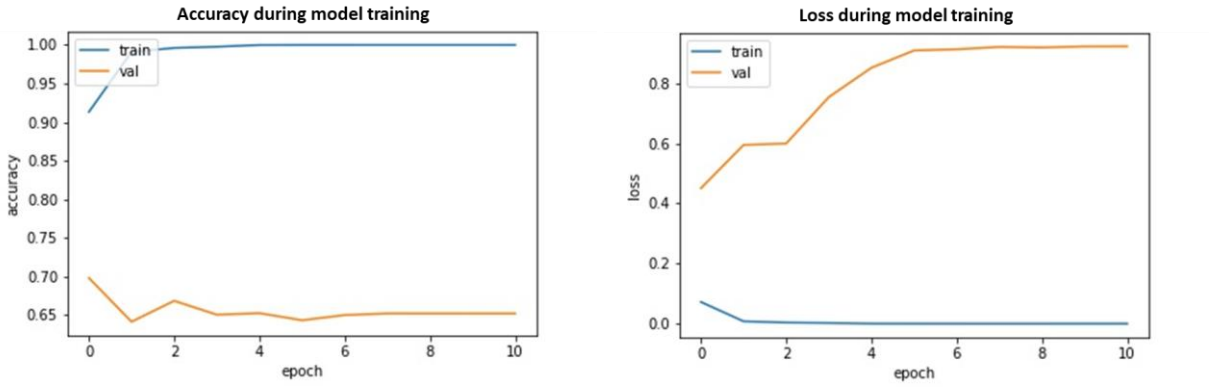

Figure 7.13: *Accuracy and Loss evolution of final model during training, using testing dataset as validation set for depicting the behavior of the model on testing data (Training Sample = 200k, Validation Sample = 100k obtained from sampling with replacement for testing set, Validation Accuracy in best epoch = 0.698 in first epoch).*

In summary, as observed in figure 7.13, the model performs in testing set in the same way that with validation set, which confirms the usefulness of the learned weights when applied to unseen data.

Figure 7.14 presents some of the time sequences that were correctly predicted by final model:

Seq_Id	Question	Actual Time Seq	Predicted Time Seq	Similarity Score
V002	Cuánto ganamos con la categoría de no carbonatados en 2018	2018	2018	1
V051	Cuál es la tendencia de ventas de gatorade en los últimos 3 años	LAST 3 YEARS	LAST 3 YEARS	1
D001	ventas totales en el mes de marzo de la marca gatorade	03_MARZO	03_MARZO	1
D003	cual es el promedio de ventas de los refrescos en los ultimos 2 meses	LAST 2 MONTHS	LAST 2 MONTHS	1
D004	que producto presenta las ventas en los ultimos 6 meses	LAST 6 MONTHS	LAST 6 MONTHS	1
D005	para canada dry y sangria casera, venta de productos en el mes de noviembre del 2019?	2019 11NOVIEMBRE	2019 11NOVIEMBRE	1
D008	cuales han sido los desempeños de las marcas en el mes de septiembre	09_SEPTIEMBRE	09_SEPTIEMBRE	1
D012	promedio de ventas de los ultimos 4 meses	LAST 4 MONTHS	LAST 4 MONTHS	1
D014	cuantas ventas tuvieron las bebidas no carbonatas en el mes de enero	01_ENERO	01_ENERO	1
D046	ventas por marca en julio	07_JULIO	07_JULIO	1
D050	ventas sangria casera y otras bebidas en febrero del 2019	2019 02FEBRERO	2019 02FEBRERO	1
D053	Cuál es la tendencia de las ventas de agua enbomtellada en 2019?	2019	2019	1
D059	Cuántas son las ventas de refrescos por marca en el último mes?	LAST MONTH	LAST MONTH	1
D064	Cuál es el valor de las compras de cada marca de refrescos en el último año?	LAST YEAR	LAST YEAR	1
D066	ventas por producto durante el invierno?	04INVIERNO	04INVIERNO	1

Figure 7.14: *Sample of 15 questions correctly translated to time sequences by final model in the testing dataset.*

While figure 7.15 presents some of the time sequences translated incorrectly:

Seq_Id	Question	Actual Time Seq	Predicted Time Seq	Similarity Score
V020	cuantos ventas tenemos en promedio cada mes independientemente de la categoría durante 2018	2018 EVERY MONTH	2018 04_ABRIL	0.33
V052	Dime las ventas por semana de los últimos tres meses	LAST 3 MONTHS EVERY WEEK	LAST 13 MONTHS	0.4
V076	compara no carbonatados vs refresco por mes 2019	2019 EVERY MONTH	2019	0.33
V078	Cuánto vendió almacen norte en los últimos tres meses y almacén sur	LAST 3 MONTHS	LAST DAY	0.33
V104	ingreos total de agua para primer cuarto	01o QUARTER	FROM LAST 06_JUNIO TO TODAY	0
D002	cual ha sido el promedio de ventas del agua en el mes de febrero del 2018	2018 02_FEBRERO	FROM 02_FEBRERO EVERY MONTH	0.25
D010	ventas de marcas entre mayo y diciembre del 2018	2018 05_MAYO 12_DICIEMBRE	2019 FROM 05_MAYO TO 12_DICIEMBRE	0.4
D018	cantidad de refrescos vendidos por sucursal y por la semana del 10 al 18 de junio	06_JUNIO FROM 10 TO 18	LAST YEAR 06_JUNIO 18 NEXT YEAR	0.33
D025	cuanto es el total de petit vendido el 1 de enero	01_ENERO 1	FROM 01_ENERO 1 TO 01_ENERO 1	0.33
D026	el 20 de julio cuantas ventas de mix se hicieron en la sucursaal este	07_JULIO 20	LAST QUARTER 07_JULIO	0.33
D030	cual es la media de productos no carbonatados vendidos entre el 28 de junio y el de julio del 2018	2018 FROM 06_JUNIO 28 TO 07_JULIO	2018 07_JULIO	0.33
D037	cantidad de bebidas no carbonatadas vendidas entre el 5 de mayo y el 3 de octubre	FROM 05_MAYO 5 TO 10_OCTUBRE 3	LAST 05_MAYO 3 YEARS	0.33
D039	cual es la marca menos vendida el 26 de enero del 2019	2019 01_ENERO 26	2019 12_DICIEMBRE 26 TO PLUS 26 MONTHS	0.29
D045	como se vendio gatorade por dias del mes de abril	04_ABRIL EVERY DAY	FROM 04_ABRIL TO 04_ABRIL	0.25
D052	Cuáles son las ventas acumuladas trimestrales en el año 2019?	2019 EVERY QUARTER	2019	0.33

Figure 7.15: *Sample of 15 questions incorrectly translated to time sequences by final model in the testing dataset.*

7.9. Conclusions of Experiments

As observed in experiments, the accuracy of the translation of questions to time sequences is better than random for validation and test data, proving that DL models can learn to summarize the temporal information contained in a natural language question into a normalized sequence (fig. 7.14).

Although the models are clearly overfitting during the very early moments of training across all experiments, no matter the batch size, dropout rate or DL architecture used, the Data Augmentation method by Paraphrases Substitution is providing enough variability in the training data for reaching a 0.65 accuracy in the validation data, 0.17 above the baseline. Moreover, this accuracy has prevailed when evaluating final model with testing data. These two facts evidence the usefulness of paraphrases substitution for providing data to train models in the translation from questions to time sequences, that can generalize part of the learning to unseen datasets. However, this method still needs to provide greater variability for training datasets, since the experiments are appointing to the presumption that training variability s not enough to represent the whole variability of unseen data.

Clustering techniques offer an opportunity for efficiently characterizing the question examples that current model and augmentation method are not being able to predict with sufficient accuracy. Once the clusters are defined, solutions can go from building specific models for such cluster up to increasing the number of examples for clusters with low accuracy, by developing augmentation methods to generate new examples satisfying the features that describe each cluster. The clustering used in this experimentation lacked from enough quality to contribute to the success of the prediction models; however, it is worth to try again with more careful analysis.

Chapter 8

8. CONCLUSIONS

8.1. Accomplishment of Research Objectives

As result of this work, we have reached all the research objectives stated at chapter 1:

- **Research Objective 1.1.** We developed TimeSeq, a standard defined by a set of rules (chapter 5) for easily annotating the temporal information contained in natural language questions as normalized time sequences, which contain all temporal constraints necessary to perform a correct search. Currently this standard applies to the context of questions made about the content of a transactional database; but could grow to other domains.

- **Research Objective 1.2.** We demonstrated with our experiments (chapter 7), that deep learning models can learn to summarize the temporal information contained in a natural language question into a normalized sequence.

- **Research Objective 2.1.** We developed a process for augmenting examples of pairs of questions and time sequences, based in the substitution of paraphrases corresponding to 3 ontologies we integrated: (1) Ontology of Paraphrases for Temporal Expressions, (2) Ontology of Paraphrases for General Business Database Querying, and (3) Ontology of Paraphrases for Specific Industry Domain (Chapter 6).

- **Research Objective 2.2.** We developed an algorithm (chapter 6) to perform the data augmentation, by creating new pairs of questions and time sequences without affecting the fidelity of the temporal information, as result of controlled combinations of paraphrases substituted at original pairs and at the new pairs generated.

- **Research Objective 2.3.** We trained two DL architectures (Transfomer and Bi-RNN with Attention) capable of translating questions into normalized time sequences (Chapter 7).

- **Research Objective 2.4.** We evidenced with our experiments (chapter 7), the usefulness of paraphrases substitution for providing data to train models in the translation from questions to time sequences, that can generalize part of the learning to unseen datasets. However, training

variability is not enough to represent the whole variability of unseen data, i.e., the models generalize what they can with the variability present in the data they were trained.

8.2. Main Contributions

Our main contribution with this work is “TimeSeq”, our proposition for storing temporal information as normalized token sequences, which can take advantage of DL models that perform seq-to-seq translation for automatically normalizing relative time information contained in common questions about business databases. We have evidenced that this approach is feasible; but we still require more data and analysis to get models that learn to summarize these time sequences with high accuracy for unseen data.

Although our data augmentation method based in paraphrases substitution still needs to improve, it provides a relevant contribution for developing new data augmentation methods that enable the use of DL models in cases where the number of examples is too few for training.

Our “Ontology of Paraphrases for Temporal Expressions” and our “Ontology of Paraphrases for General Business Database Querying” constitute a first attempt for integrating useful resources in Spanish Language that can be used for supporting Data Augmentation through paraphrases substitution and other applications across distinct industries and domains.

8.3. Learnings

Our main learnings during this work derives from the implementation of TimeSeq and the Data Augmentation method. When we started with the definition of TimeSeq standard, we try to create strict rules requiring people to make explicit all temporal information contained in questions and trying to avoid that distinct operators were use for describing same region in the space of temporal information. However, we realized that for facilitating the translation task, we would need to create sequences derived from natural language sequences, and not to create rigid templates or schemas for extracting information. For this reason, we changed our approach to use “controlled ambiguity” in the creation of time sequences that looked more like natural language; but keeping a limited number of operators and more robust rules, based in common knowledge, that allow the extraction

of temporal constraints without ambiguity. For example: “compare sales to date with respect to same period the previous year” initially was annotated as “CURRENT YEAR TODATE VS LAST YEAR TODATE”; but now it is annotated as “VS LAST YEAR TODATE” or “VS PREVIOUS YEAR TODATE”, where the current year to date is implicit to the left of the “VS” operator, by applying a common knowledge rule.

We started by defining the rules of TimeSeq, the Ontologies and the Data Augmentation Algorithm independently from each other; however, this derived into poor performance of the augmentation process and into many cycles of adjustments. When planning changes into data augmentation process, the three elements and the way they interact need to be considered the whole time to avoid low performance of the augmentation, which is hard to detect since most of the time there are no alerts.

8.4. Future Work

We propose to continue working to optimize the current Data Augmentation method and the capability of the models to predict TimeSeq, before trying to expand their application and the application of current data augmentation method to other domains or tasks.

We need to evaluate the impact of non-contextual and contextual embeddings in the generalization of accuracy to unseen data.

We have to integrate an additional measure of accuracy, for comparing the results of the query restricted by actual and predicted time sequences, instead of only evaluating that both sequences are written the same way.

It is convenient we explore the total combinations that can be reached for all pairs of sequences, leaving behind the constraint on the maximum number of new pairs that can be generated for an input pair, since this can contribute to increase variability in training data.

We have to get better clusters to characterize the question examples that current model and augmentation method are not being able to predict with sufficient accuracy, as well as clusters that can render better results if training their own translation models.

We can explore syntactic parsing and dependencies as medium for altering the position of paraphrases in questions, for incorporating additional variability in augmented training data.

Bibliography

- Steven Bethard and Jonathan Parker. (2016). A semantically compositional annotation scheme for time normalization. In Lrec, volume 2016, pages 3779– 3786.

- Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk and Yoshua Bengio. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In EMNLP.

- Francois Chollet and others. (2015). Keras. https://keras.io/guides/

- Ismael Gámez. (2013). SNL2SQL: Conversión de consultas en SQL al idioma Español.

- Juri Ganitkevitch and Chris Callison-Burch. (2014). The Multilingual Paraphrase Database. Proceedings of The 9th edition of the Language Resources and Evaluation Conference. http://cis.upenn.edu/~ccb/publications/ppdb-multilingual.pdf

- L. Ferro, L., Gerber, I. Mani, B. Sundheim and G. Wilson. (2005). Tides 2005 standard for the annotation of temporal expressions. Technical report.

- Sepp Hochreiter & Jürgen Schmidhuber. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

- Minh-Thang Luong, Hieu Pham and Christopher D. Manning. (2015). Effective approaches to attention based neural machine translation. In EMNLP.

- J. Pustejovsky, K. Lee, H. Bunt and L. Romary. (2010). Iso-timeml: An international standard for semantic annotation. In Nicoletta Calzolari (Conference Chair), et al., editors, Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, may. European Language Resources Association (ELRA).

- M. Schuster & K. K. Paliwal. (1997). "Bidirectional recurrent neural networks," in IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681. doi: 10.1109/78.650093

- Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin. (2017). Attention Is All You Need. https://arxiv.org/abs/1706.03762

- Tao Yu,Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang and Dragomir Radev. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task https://yale-lily.github.io/spider

- Victor Zhong, Caiming Xiong, and Richard Socher. (2017). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. https://github.com/salesforce/WikiSQL