Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Adaptação para a Língua Portuguesa de Ferramentas Text-to-SQL

Antônio Carlos Santos de Lima

mac 499— Trabalho de Formatura Supervisionado

Supervisora: Profª. Drª. Kelly Rosa Braghetto

São Paulo
26 de Setembro de 2022

Resumo

Antônio Carlos Santos de Lima. **Adaptação para a Língua Portuguesa de Ferramentas Text-to-SQL**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

A maior parte dos dados persistidos por sistemas de software é armazenada em bancos de dados relacionais, que são consultados e atualizados por meio de comandos especificados em SQL (*Structured Query Language*), a linguagem declarativa padrão para esse tipo de banco de dado. Realizar consultas em um banco de dados usando SQL exige conhecimento prévio tanto da estrutura do banco quanto da linguagem de consulta, o que torna a tarefa árdua para usuários não técnicos. A fim de mitigar esse problema, pesquisadores desenvolveram mecanismos nomeados *Text-to-SQL*, para converter uma sentença em linguagem natural para um comando SQL. O NaLIR e o GLAMORISE são duas dessas ferramentas de software que, originalmente, utilizam apenas sentenças em Inglês. Nesta monografia, essas duas ferramentas foram adaptadas para serem utilizadas com sentenças em língua portuguesa. A fim de testar os sistemas adaptados, usaram-se dados da Fundação SEADE referentes a casos de COVID-19 e especificaram-se sentenças em Português relacionadas a esses dados. De modo geral, os comandos SQL obtidos da conversão automática foram condizentes com as sentenças usadas nos experimentos.

Palavras-chave: text-to-sql. bancos de dados. processamento em linguagem natural. interfaces de linguagem natural.

Abstract

Antônio Carlos Santos de Lima. **Adaptation of Text-to-SQL tools to the Portuguese language**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

Most of the data that is persisted by software systems is stored in relational databases, which are consulted and updated through commands specified in SQL (Structured Query Language), the default declarative language to that type of database. Running queries in a database by using SQL requires prior knowledge of both the database structure and the query language, which makes that task hard for non-technical users. In order to mitigate that problem, researchers have developed mechanisms named Text-to-SQL, to convert a natural language sentence into a SQL query. NaLIR and GLAMORISE are two of these software tools that originally use only sentences in English. In this capstone project, these tools have been adapted in order to be used with sentences in Portuguese. In order to test the adapted systems, the SEADE Foundation's data related to COVID-19 cases has been used, and Portuguese sentences related to that data have been specified and converted. On the whole, the SQL queries obtained from the automatic conversions have been consistent with the sentences that have been used in the experiments.

Keywords: text-to-sql. databases. natural language processing. natural language interfaces.

Sumário

1 Introdução **1**
1.1 Contexto . . . 1
1.2 Motivação . . . 2
1.3 Objetivos . . . 2
1.4 Resumo dos resultados . . . 3

2 Fundamentos **4**
2.1 Processamento de linguagem natural . . . 4
2.1.1 Tokenização . . . 4
2.1.2 Análise léxica . . . 5
2.1.3 Análise semântica . . . 7
2.2 Bancos de dados relacionais . . . 7
2.2.1 Conceitos pertinentes a bancos de dados . . . 7
2.3 SQL . . . 9
2.3.1 SELECT-FROM-WHERE . . . 9
2.3.2 Ausência de cláusula WHERE e uso de asterisco . . . 9
2.3.3 DISTINCT e ALL . . . 10
2.3.4 LIKE . . . 10
2.3.5 ORDER BY . . . 10
2.3.6 Agregações . . . 11
2.3.7 GROUP BY . . . 11

3 Ferramentas **13**
3.1 NaLIR . . . 13
3.2 GLAMORISE . . . 15

4 Desenvolvimento do trabalho: Adaptações **17**
4.1 Inclusão de arquivos de configuração . . . 17
4.2 Alteração de arquivos . . . 18

4.3 Uso geral do NaLIR e do GLAMORISE 21

5 Experimentos e resultados **22**
5.1 Dados 22
5.1.1 Estrutura 22
5.1.2 Banco de dados 23
5.2 Arquivo de configuração para um BD específico 23
5.3 Arquivos para teste dos sistemas adaptados 24
5.4 Consultas 25
5.4.1 Sem agregações e sem cláusula “WHERE” 25
5.4.2 Sem agregações e com cláusula “WHERE” 26
5.4.3 Agregações sem agrupamentos e sem cláusula “WHERE” 29
5.4.4 Agregações sem agrupamentos e com cláusula “WHERE” 32
5.4.5 Agregações com agrupamentos e sem cláusula “WHERE” 35
5.4.6 Agregações com agrupamentos e com cláusula “WHERE” 37
5.5 Discussão 41

6 Conclusão **42**
6.1 Disciplinas relacionadas ao trabalho 42
6.2 Trabalhos futuros 43

Apêndices

A Figuras: relação de dependência e resultado do processamento no NaLIR / GLAMORISE **44**

Referências **68**

Lista de Figuras

2.1 Exemplo de uma gramática (adaptado de Barbosa *et al.*, 2017 *apud* Moura, 1996) . . . 5
2.2 Árvore sintática da frase *"Na semana passada, o número de casos positivos de COVID-19 aumentou em Campinas."* . . . 6
2.3 Atributos e tuplas da relação "CASOS", adaptada do banco de dados da Fundação SEADE . . . 8
2.4 Resultado da consulta com a cláusula "GROUP BY" . . . 12

3.1 Árvore de consulta para o comando SQL "SELECT nome_munic, raca_cor FROM casos WHERE nome_munic = "São Paulo" OR raca_cor = "parda". σ indica uma operação "SELECT" e π indica uma operação "PROJECT", em que apenas determinadas colunas de uma relação são selecionadas . . . 15
3.2 Arquitetura do NALIR (Li e Jagadish, 2014) . . . 15
3.3 Arquitetura do GLAMORISE (Novello, 2021) . . . 16

4.1 Trecho de código do arquivo "glamorise_nalir_adapted.json". Nesse trecho, informa-se um padrão em Português em que um substantivo comum ou próprio geralmente é sucedido por um ou nenhum adjetivo . . . 18
4.2 Trecho de código do arquivo "nalir_tokens_adapted.xml". Nesse trecho, informa-se que determinados termos ("qual", "quais", "quantos", etc.) são verbos classificados como tokens de comando (command token) . . . 18
4.3 Trecho de código do arquivo "glamorise.py". Nesse trecho, ocorre a formação do comando SQL final. Para isso, reúnem-se os elementos identificados como parte das cláusulas "SELECT", "FROM", "GROUP BY", "HAVING" e "ORDER BY". Não necessariamente todas essas cláusulas aparecem no comando SQL final . . . 19

4.4 Trecho de código do arquivo “node_mapper.py”. Nesse trecho, os nós da árvore de análise são categorizados, alguns nós considerados inúteis são removidos e estabelecem-se correspondências entre nós e atributos/valores contidos no banco de dados 19
4.5 Trecho de código do arquivo “stanford_parser.py”. Nesse trecho, utilizam-se funções auxiliares para a construção de uma tabela auxiliar para a formação da árvore de análise 20

5.1 Trecho de código do arquivo “glamorise_interface_nalir_seade.json”. Nesse trecho, mostra-se a correspondência entre palavras (ou termos compostos separados por um “_”) em Português e os atributos da tabela “casos” no banco “seade” 24
5.2 Trecho de código do arquivo “main_nalir_seade_single_nlq.py”. Nesse trecho, um objeto da classe “GlamoriseNlidb” é criado e, na função da linha 75, esse objeto é utilizado para obter o comando SQL de uma sentença em Português, localizada na posição “i” da lista “nlqs”. Certas partes do resultado desse processamento foram salvas em arquivos utilizados durante os experimentos 24
5.3 Diagrama de dependências da consulta “Retorne todas as idades dos casos onde o município é São Paulo”, mencionada na seção 5.4.2 27
5.4 Resultado simplificado do processamento da consulta “Retorne todas as idades dos casos onde o município é São Paulo”, mencionada na seção 5.4.2 27
5.5 Diagrama de dependências da consulta “Retorne todas as idades dos casos do município de São Paulo”, mencionada na seção 5.4.2 28
5.6 Resultado simplificado do processamento da consulta “Retorne todas as idades dos casos do município de São Paulo”, mencionada na seção 5.4.2 . 28
5.7 Diagrama de dependências da consulta “Devolva quantos óbitos existem em casos”, mencionada na seção 5.4.3 30
5.8 Resultado simplificado do processamento da consulta “Devolva quantos óbitos existem em casos”, mencionada na seção 5.4.3 30
5.9 Diagrama de dependências da consulta “Devolva quantos casos há”, mencionada na seção 5.4.3 31
5.10 Resultado simplificado do processamento da consulta “Devolva quantos casos há”, mencionada na seção 5.4.3 31
5.11 Diagrama de dependências da consulta “Devolva quantos óbitos do tipo positivo onde o município é São Paulo”, mencionada na seção 5.4.4 33

5.12 Resultado simplificado do processamento da consulta “Devolva quantos óbitos do tipo positivo onde o município é São Paulo”, mencionada na seção 5.4.4 33
5.13 Diagrama de dependências da consulta “Devolva quantos óbitos do tipo positivo há no município de São Paulo”, mencionada na seção 5.4.4 . . . 34
5.14 Resultado simplificado do processamento da consulta “Devolva quantos óbitos do tipo positivo há no município de São Paulo”, mencionada na seção 5.4.4 34
5.15 Diagrama de dependências da consulta “Devolva quantos casos há por município”, mencionada na seção 5.4.5 36
5.16 Resultado simplificado do processamento da consulta “Devolva quantos casos há por município”, mencionada na seção 5.4.5 36
5.17 Diagrama de dependências da consulta “Devolva quantos óbitos do tipo positivo há no município de São Paulo por raça”, mencionada na seção 5.4.6 38
5.18 Resultado simplificado do processamento da consulta “Devolva quantos óbitos do tipo positivo há no município de São Paulo por raça”, mencionada na seção 5.4.6 38
5.19 Diagrama de dependências da consulta “Devolva quantos diagnóstico há agrupados por raça e por sexo e por diagnóstico”, mencionada na seção 5.4.6 40
5.20 Resultado simplificado do processamento da consulta “Devolva quantos diagnóstico há agrupados por raça e por sexo e por diagnóstico”, mencionada na seção 5.4.6 40

A.1 Diagrama de dependências da consulta “Retorne todos os municípios em casos”, mencionada na seção 5.4.1 44
A.2 Resultado simplificado do processamento da consulta “Retorne todos os municípios em casos”, mencionada na seção 5.4.1 45
A.3 Diagrama de dependências da consulta “Devolva a máxima idade em casos”, mencionada na seção 5.4.3 46
A.4 Resultado simplificado do processamento da consulta “Devolva a máxima idade em casos”, mencionada na seção 5.4.3 46
A.5 Diagrama de dependências da consulta “Devolva a mínima idade em casos”, mencionada na seção 5.4.3 47
A.6 Resultado simplificado do processamento da consulta “Devolva a mínima idade em casos”, mencionada na seção 5.4.3 47
A.7 Diagrama de dependências da consulta “Devolva as idades somadas em casos”, mencionada na seção 5.4.3 48

A.8 Resultado simplificado do processamento da consulta “Devolva as idades somadas em casos”, mencionada na seção 5.4.3 48
A.9 Diagrama de dependências da consulta “Devolva a idade média em casos”, mencionada na seção 5.4.3 49
A.10 Resultado simplificado do processamento da consulta “Devolva a idade média em casos”, mencionada na seção 5.4.3 49
A.11 Diagrama de dependências da consulta “Devolva quantos óbitos do tipo positivo existem em casos”, mencionada na seção 5.4.4 50
A.12 Resultado simplificado do processamento da consulta “Devolva quantos óbitos do tipo positivo existem em casos”, mencionada na seção 5.4.4 . . 50
A.13 Diagrama de dependências da consulta “Devolva a máxima idade no município de São Paulo”, mencionada na seção 5.4.4 51
A.14 Resultado simplificado do processamento da consulta “Devolva a máxima idade no município de São Paulo”, mencionada na seção 5.4.4 51
A.15 Diagrama de dependências da consulta “Devolva a mínima idade no município de São Paulo”, mencionada na seção 5.4.4 52
A.16 Resultado simplificado do processamento da consulta “Devolva a mínima idade no município de São Paulo”, mencionada na seção 5.4.4 52
A.17 Diagrama de dependências da consulta “Devolva as idades somadas no município de São Paulo”, mencionada na seção 5.4.4 53
A.18 Resultado simplificado do processamento da consulta “Devolva as idades somadas no município de São Paulo”, mencionada na seção 5.4.4 53
A.19 Diagrama de dependências da consulta “Devolva a idade média no município de São Paulo”, mencionada na seção 5.4.4 54
A.20 Resultado simplificado do processamento da consulta “Devolva a idade média no município de São Paulo”, mencionada na seção 5.4.4 54
A.21 Diagrama de dependências da consulta “Devolva a máxima idade por município”, mencionada na seção 5.4.5 55
A.22 Resultado simplificado do processamento da consulta “Devolva a máxima idade por município”, mencionada na seção 5.4.5 55
A.23 Diagrama de dependências da consulta “Devolva a mínima idade por município”, mencionada na seção 5.4.5 56
A.24 Resultado simplificado do processamento da consulta “Devolva a mínima idade por município”, mencionada na seção 5.4.5 56
A.25 Diagrama de dependências da consulta “Devolva as idades somadas por município”, mencionada na seção 5.4.5 57

A.26 Resultado simplificado do processamento da consulta “Devolva as idades somadas por município”, mencionada na seção 5.4.5 57
A.27 Diagrama de dependências da consulta “Devolva a idade média por município”, mencionada na seção 5.4.5 58
A.28 Resultado simplificado do processamento da consulta “Devolva a idade média por município”, mencionada na seção 5.4.5 58
A.29 Diagrama de dependências da consulta “Devolva a máxima idade no município de São Paulo por raça”, mencionada na seção 5.4.6 59
A.30 Resultado simplificado do processamento da consulta “Devolva a máxima idade no município de São Paulo por raça”, mencionada na seção 5.4.6 . . 59
A.31 Diagrama de dependências da consulta “Devolva a mínima idade no município de São Paulo por raça”, mencionada na seção 5.4.6 60
A.32 Resultado simplificado do processamento da consulta “Devolva a mínima idade no município de São Paulo por raça”, mencionada na seção 5.4.6 . . 60
A.33 Diagrama de dependências da consulta “Devolva as idades somadas no município de São Paulo por raça”, mencionada na seção 5.4.6 61
A.34 Resultado simplificado do processamento da consulta “Devolva as idades somadas no município de São Paulo por raça”, mencionada na seção 5.4.6 61
A.35 Diagrama de dependências da consulta “Devolva a idade média no município de São Paulo por raça”, mencionada na seção 5.4.6 62
A.36 Resultado simplificado do processamento da consulta “Devolva a idade média no município de São Paulo por raça”, mencionada na seção 5.4.6 . 62
A.37 Diagrama de dependências da consulta “Devolva quantos óbitos do tipo positivo há no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 63
A.38 Resultado simplificado do processamento da consulta “Devolva quantos óbitos do tipo positivo há no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 63
A.39 Diagrama de dependências da consulta “Devolva a máxima idade no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 64
A.40 Resultado simplificado do processamento da consulta “Devolva a máxima idade no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 64
A.41 Diagrama de dependências da consulta “Devolva a mínima idade no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 65
A.42 Resultado simplificado do processamento da consulta “Devolva a mínima idade no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 65

A.43 Diagrama de dependências da consulta “Devolva as idades somadas no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 . . . 66
A.44 Resultado simplificado do processamento da consulta “Devolva as idades somadas no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 . . . 66
A.45 Diagrama de dependências da consulta “Devolva a idade média no município de São Paulo por diagnóstico”, mencionada na seção 5.4.6 . . . 67
A.46 Resultado simplificado do processamento da consulta “Devolva a idade média no município de São Paulo por diagnóstico”, 5.4.6 . . . 67

Lista de Tabelas

5.1 Dicionário para os atributos da relação “CASOS” 23

Capítulo 1

Introdução

1.1 Contexto

A maior parte dos dados persistidos por sistemas de software é armazenada em bancos de dados relacionais. Bancos de dados relacionais são consultados e atualizados por meio de comandos especificados em SQL (*Structured Query Language*), a linguagem declarativa padrão para esse tipo de bancos de dados.

Realizar consultas em um banco de dados usando SQL exige conhecimento prévio tanto da estrutura do banco quanto da linguagem de consulta, o que torna a tarefa árdua para usuários não técnicos.

A fim de mitigar esse problema, pesquisadores desenvolveram mecanismos, nomeados *Text-to-SQL*, para converter uma sentença em linguagem natural para um comando SQL. Por exemplo, supondo que, em um banco de dados, exista a tabela "Autor" e um campo "nome", então para a sentença

> *Retorne todos os autores cujo nome é "Antônio"*

o comando SQL correspondente deve ser

> *SELECT * FROM autor WHERE nome = "Antônio"*

Text-to-SQL envolve técnicas e modelos das áreas de Processamento de Linguagem Natural e Aprendizado de Máquina, em especial modelos do tipo Seq2Seq, arquiteturas da forma codificador-decodificador, aprendizagem por reforço e mecanismo de atenção, muitas vezes combinados entre si numa arquitetura complexa.

Dentre os trabalhos relacionados com ferramentas *Text-to-SQL*, há aquele realizado por Li e Jagadish, 2014, em que é proposta uma Interface de Linguagem Natural para Banco de Dados Relacionais (*Natural Language Interface for Relational databases*, em Inglês); e o trabalho de Novello, 2021, no qual é apresentada a ferramenta GLAMORISE, para processar sentenças em linguagem natural com agregação, que o NaLIR (ou outra interface similar) geralmente não converte em comando SQL de forma satisfatória.

Em uma agregação, sumarizam-se informações de múltiplas tuplas em uma única tupla

(Elmasri e Navathe, 2015). Por exemplo, utilizando a mesma tabela "Autor" do exemplo acima e supondo que existam 10 registros nessa tabela, o comando SQL

SELECT COUNT(*) FROM autor

contém a operação COUNT, que retorna a quantidade de registros na tabela, que, nesse caso, é 10.

Há ferramentas que dependem de grandes volumes de dados de treinamento para criar modelos de conversão para um banco de dados em particular, como, por exemplo, o sistema de resposta a perguntas (*question answering system*) (Liao *et al.*, 2020); o BERT (*Bidirectional Encoder Representations from Transformers*) (Devlin *et al.*, 2018); e o Raciocínio Neural de Resposta a Perguntas (*Neural Question Answering Reasoner*) (José *et al.*, 2022). O NaLIR e o GLAMORISE são diferentes nesse aspecto, pois não precisam ser treinados para um banco de dados em particular.

Na ferramenta NaLIR, uma sentença em linguagem natural é dada como entrada e, após ser processada, é gerado um comando SQL, conforme a sentença dada como entrada. No sistema GLAMORISE, parte do processamento feito no NaLIR é utilizado, para lidar com possíveis agregações e retornar um comando SQL que expresse melhor a consulta original feita em linguagem natural.

1.2 Motivação

Durante o desenvolvimento deste trabalho de conclusão de curso, não foram encontradas ferramentas para a conversão de consultas em Português para comandos SQL que não dependessem de treinamento de modelos.

Além disso, as ferramentas NaLIR e GLAMORISE foram desenvolvidas para processar apenas sentenças em Inglês, utilizando processadores de texto específicos para a língua inglesa e contendo várias referências a termos da língua inglesa embutidas no código (*hard-coded*). Portanto, a implementação original dessas ferramentas não mapeia de forma apropriada consultas em português.

1.3 Objetivos

O objetivo principal deste trabalho foi adaptar a NaLIR e o GLAMORISE, para serem utilizados em consultas e bancos de dados em língua portuguesa.

Os objetivos específicos do trabalho foram:

- Estudar o NaLIR e o GLAMORISE, desenvolvidos respectivamente por Li e Jagadish, 2014 e Novello, 2021;
- Adaptar as propostas de Li e Jagadish, 2014 e Novello, 2021 para a utilização com sentenças em língua portuguesa;
- Avaliar as adaptações desenvolvidas usando diferentes tipos de consultas sobre um banco de dados real.

1.4 Resumo dos resultados

Realizaram-se experimentos com sentenças em língua portuguesa, a fim de verificar a qualidade dos comandos SQL gerados. Para isso, utilizaram-se dados disponíveis em SEADE, 2022a e as sentenças foram determinadas com base nesses dados.

Para alguns formatos de sentenças em língua portuguesa, foi possível obter comandos SQL satisfatórios, que puderam ser utilizados em consultas de dados disponibilizados pela Fundação SEADE, relacionados a ocorrências de COVID-19 no Brasil. Neste trabalho, as consultas geralmente possuem um verbo no modo imperativo, seguido de algum atributo ou quantidade de interesse (número de pacientes, número de mortes), seguido por alguma condição, como, por exemplo, o nome de uma cidade, o sexo ou a cor de um paciente. No capítulo sobre os resultados, mostra-se o mapeamento das consultas feitas e, para variantes dessas consultas, a qualidade do mapeamento variou.

A fim de obter os resultados mostrados neste trabalho, foram feitas algumas alterações no NaLIR e no GLAMORISE, como, por exemplo:

- Utilização da biblioteca *Spacy* como substituta ou complemento à biblioteca *NLTK*;
- Substituição de termos em Inglês contidos em algumas partes do código do projeto original;
- Implementação de algoritmo que realiza operações originalmente feitas pela biblioteca *Stanford Parser*.

Capítulo 2

Fundamentos

Neste capítulo, serão apresentados conceitos sobre processamento de linguagem natural, banco de dados relacionais e linguagem SQL. Esses conceitos são importantes para a compreensão dos capítulos seguintes desta monografia.

2.1 Processamento de linguagem natural

O processamento de linguagem natural é uma área da Computação cujo objetivo é a extração de representação e de significados mais completos de textos livres escritos em linguagem natural (Barbosa *et al.*, 2017 *apud* Indurkhya e Damerau, 2010), que, por sua vez, é a linguagem utilizada por humanos no dia a dia para a comunicação com outros humanos (Thanaki, 2017), como, por exemplo, Inglês, Espanhol, Português, Alemão, entre outras.

O processamento de um texto em linguagem natural geralmente é dividido em algumas etapas, que serão descritas a seguir.

2.1.1 Tokenização

Nessa etapa, quebra-se a sequência de caracteres de um texto localizando o início e o fim de cada palavra nesse texto. Há tipos diferentes de tokenização, dependendo do tipo de linguagem utilizado (Barbosa *et al.*, 2017).

Em linguagens delimitadas por espaço, indicam-se os limites de cada palavra por um espaço em branco, como ocorre em muitas linguagens europeias. As sequências de caracteres delimitadas não são necessariamente os *tokens* requeridos para processamento adicional, devido às ambiguidades dos sistemas de escrita e também ao conjunto de convenções de tokenização utilizadas por diferentes aplicações (Barbosa *et al.*, 2017 *apud* Indurkhya e Damerau, 2010). Nesse tipo de linguagem, as ambiguidades geralmente acontecem no uso dos sinais de pontuação, os quais podem exercer funções diferentes em uma mesma frase.

Em linguagens não segmentadas, como, por exemplo, o chinês e o tailandês, as palavras não possuem um limite no texto, necessitando, portanto, de informações léxicas e

morfológicas complementares para a tokenização.

Considere o texto a seguir:

> *Devolva o número de casos em S. Bernardo do Campo se o número de casos for maior que 1.000.*

Usa-se o ponto final de formas diferentes nessa frase: como parte de uma abreviação ("S.", por exemplo), como fim de uma frase e como separador de classes numéricas ("1.000", por exemplo). Por causa disso, a tokenização precisa diferenciar e tratar esses casos adequadamente.

2.1.2 Análise léxica

Esse tipo de análise consiste na associação de variantes morfológicas aos seus lemas, os quais são a forma em que as palavras estão escritas nos dicionários. Utilizam-se os lemas de diferentes formas, dependendo da tarefa que precisa ser realizada pelo sistema de processamento de linguagem natural. Na análise léxica, é possível mapear a palavra até o seu lema (*parsing side*), e mapear o lema para a palavra (*morphological generation*) (Barbosa *et al.*, 2017 *apud* Indurkhya e Damerau, 2010).

É possível representar a análise sintática por meio de gramáticas e árvores sintáticas (*syntax tree* ou *parse tree*). A figura abaixo mostra um exemplo de gramática.

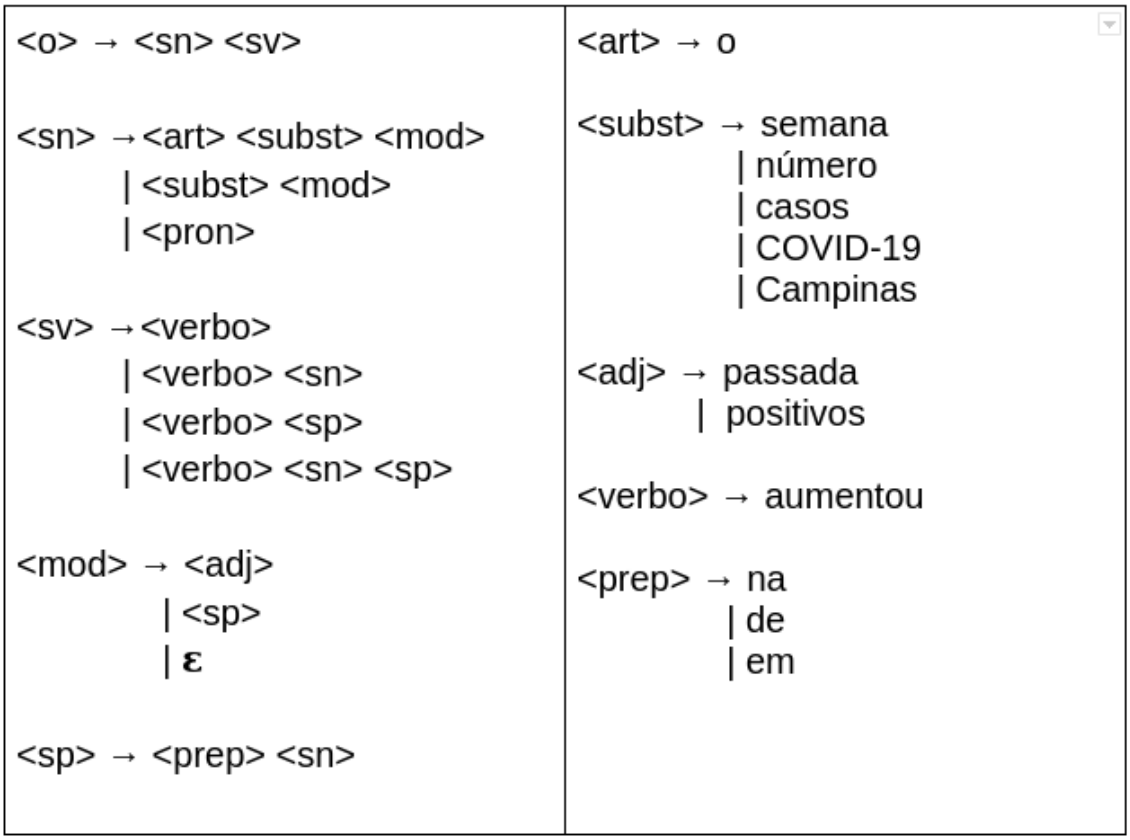


Figura 2.1: *Exemplo de uma gramática (adaptado de Barbosa et al., 2017 apud Moura, 1996)*

Na figura 2.1, a gramática reconhece a frase *"eu vi a garota com o binóculo"* e contém dois lados, em que o lado esquerdo contém os símbolos preterminais, que se decompõem sempre em uma produção, que, por sua vez, fica do lado direito, referentes aos termos léxicos (Barbosa *et al.*, 2017).

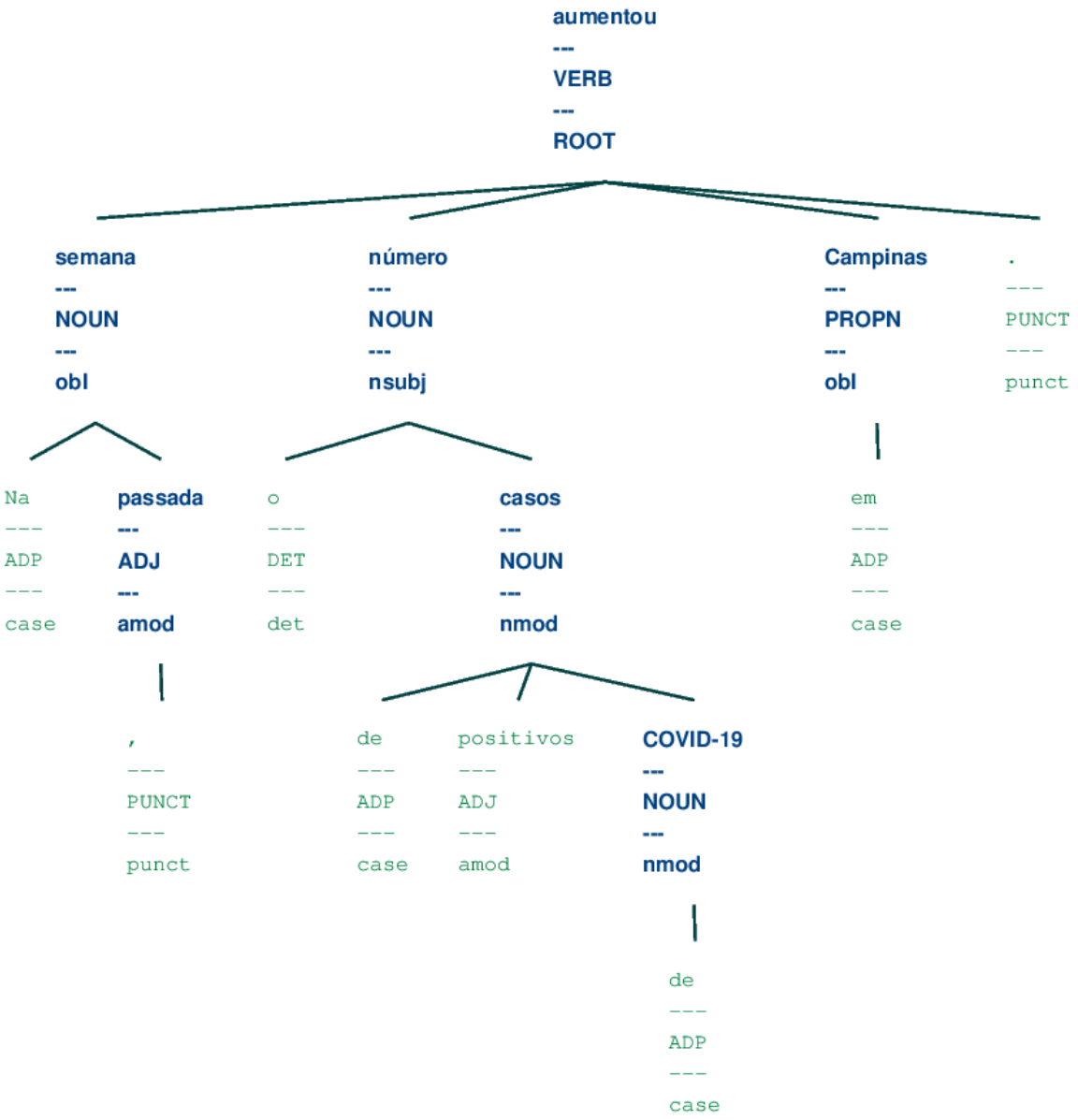


Figura 2.2: *Árvore sintática da frase "Na semana passada, o número de casos positivos de COVID-19 aumentou em Campinas."*

A árvore sintática representa todas as etapas na derivação da frase a partir do nó raiz. Cada nó interno na árvore representa a aplicação de uma regra da gramática. Na figura anterior, a árvore é formada de cima para baixo, com a raiz da árvore no topo e as folhas na parte de baixo (Barbosa *et al.*, 2017).

2.1.3 Análise semântica

Essa análise se refere à análise do significado das palavras, expressões fixadas, frases inteiras e enunciados no contexto, o que significa traduzir as expressões originais em um tipo de metalinguagem. (Barbosa *et al.*, 2017 *apud* Indurkhya e Damerau, 2010)

Nesse tipo de análise, costuma haver ambiguidades, que podem ser de três tipos:

- Léxicas: quando uma palavra pode ter mais de um significado;
- Escopais: onde quantificadores, modais ou operadores negativos podem ser utilizadas em diferentes trechos de texto;
- Referenciais: nas quais as referências de pronomes ou outras se referindo a expressões podem ser difíceis de compreender.

2.2 Bancos de dados relacionais

Um banco de dados é uma coleção de dados relacionados, que, por sua vez, são fatos conhecidos que podem ser gravados e que possuem significado implícito. Um exemplo é o conjunto de nomes, números de telefone e endereços que tipicamente são armazenados em celulares, os quais possuem o seu próprio programa de banco de dados (Elmasri e Navathe, 2015).

Um sistema de gerenciamento de banco de dados (SGBD) é um sistema computadorizado utilizado para criar e fazer manutenções em um banco de dados, além de facilitar a definição, a construção, a manipulação e o comparilhamento de banco de dados entre aplicações e usuários.

Um banco de dados pode ser acessado por uma aplicação, através do envio de uma consulta ou uma requisição. Em uma consulta, dados geralmente são recuperados do SGBD; em uma transação, é possível ler e escrever dados no banco de dados.

No modelo de dados relacional, representa-se um banco de dados como um conjunto de relações consideradas tabelas de valores, cujas linhas representam um fato correspondente a uma entidade ou relacionamento que existe na realidade e cujas colunas representam um atributo da entidade ou do relacionamento.

2.2.1 Conceitos pertinentes a bancos de dados

Cada linha nas tabelas de um banco de dados é chamada de **tupla** no modelo relacional (Elmasri e Navathe, 2015).

O nome de cada coluna é um **atributo**, que representa alguma propriedade referente a uma entidade. Por exemplo: nome ou idade de um paciente de hospital.

Uma **entidade** representa um conceito ou um objeto do mundo real. Por exemplo: paciente ou projeto de um hospital descrito no banco de dados.

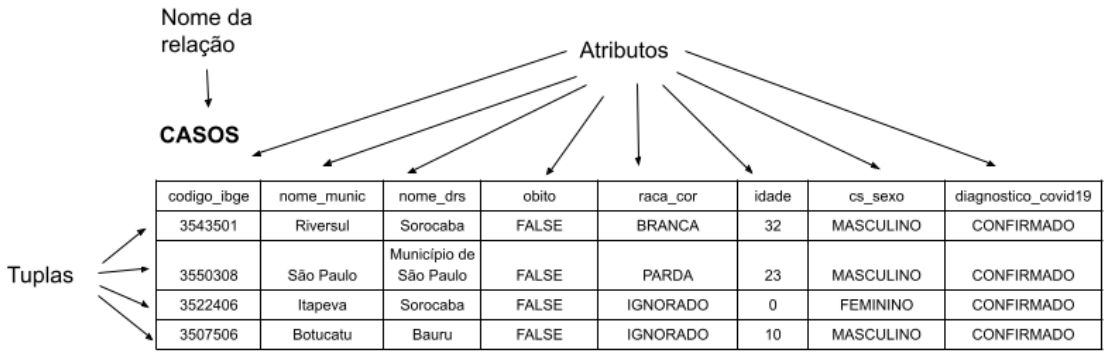
Um **relacionamento entre duas ou mais entidades** associa entidades, como, por exemplo, um relacionamento “é-atendido-em” entre um paciente e um hospital.

O tipo de dados que descreve os valores possíveis em cada coluna de uma tabela é formado por um **domínio** de valores possíveis, e esse domínio é um conjunto de valores indivisíveis.

Um **esquema de relação** R, denotado por $R(A_1, A_2, ..., A_n)$ é formado por um nome de relação R e uma lista de atributos $A_1, A_2, ..., A_n$. Um atributo A_i é o nome de uma função executada por um domínio D no esquema de relação R. D é chamado de domínio de A_i e é denotado por **dom(**A_i**)**. Utiliza-se um esquema de relação para a descrição de uma relação e R é o **nome** dessa relação.

Uma **relação** r do esquema de relação $R(A_1, A_2, ..., A_n)$, também denotada por $r(R)$, é um conjunto de n-tuplas $r = \{t_1, t_2, ..., t_m\}$, em que cada n-**tupla** t é uma lista ordenada de n valores $t =< v_1, v_2, ..., v_n >$, onde cada valor v_p, $1 \leq i \leq n$, é um elemento de dom(A_i) ou um valor nulo. A tabela de um banco de dados é uma **relação**.

A figura 2.3 mostra uma relação com o seu nome, as suas tuplas e os seus atributos. Nessa relação, registram-se dados dos casos suspeitos ou confirmados de COVID-19.



codigo_ibge	nome_munic	nome_drs	obito	raca_cor	idade	cs_sexo	diagnostico_covid19
3543501	Riversul	Sorocaba	FALSE	BRANCA	32	MASCULINO	CONFIRMADO
3550308	São Paulo	Município de São Paulo	FALSE	PARDA	23	MASCULINO	CONFIRMADO
3522406	Itapeva	Sorocaba	FALSE	IGNORADO	0	FEMININO	CONFIRMADO
3507506	Botucatu	Bauru	FALSE	IGNORADO	10	MASCULINO	CONFIRMADO

Figura 2.3: *Atributos e tuplas da relação "CASOS", adaptada do banco de dados da Fundação SEADE*

O atributo (ou o conjunto de atributos) cujo valor distingue uma tupla de todas as outras tuplas é denominado **chave primária**. Portanto, duas tuplas diferentes em todo estado de uma relação não podem possuir atributos com os mesmos valores ao mesmo tempo, se esses atributos são chave primária.

Um **esquema de banco de dados relacional** S é um conjunto de esquemas de relação $S = \{R_1, R_2, ..., R_m\}$ e um conjunto de **restrições de integridade** RI.

Um **estado de banco de dados relacional** DB de S é um conjunto de relações $DB = \{r_1, r_2, ..., r_m\}$ tal que cada r_i é um estado de R_i e tal que as relações r_i satisfazem as restrições de integridade especificadas em RI.

Um **banco de dados relacional** é composto pelo seu esquema e pelo seu estado atual.

2.3 SQL

SQL significa *Structured Query Language*, que, em Português, significa Linguagem de Consulta Estruturada (Elmasri e Navathe, 2015). É o padrão de linguagem de consulta para bancos de dados relacionais, foi criada em 1970 (Chamberlin, 2009), tornou-se padrão ANSI em 1986 (Elmasri e Navathe, 2015) e desde então vem sendo estendida com novos comandos para acompanhar a evolução dos modelos de dados e dos SGBDs (Elmasri e Navathe, 2015).

A linguagem SQL é bastante importante não apenas para bancos de dados relacionais, pois a maioria das linguagens de consulta desenvolvidas para outros tipos de bancos de dados se baseiam nela e usam sintaxe e comandos similares aos dela.

Nessa linguagem, as palavras **tabela**, **linha** e **coluna** são utilizadas para se referir aos termos *relação*, *tupla* e *atributo*, respectivamente.

SQL é uma linguagem declarativa bastante expressiva, que permite especificar uma grande variedade de operações de consultas envolvendo filtragem, junção, agrupamento e agregação de dados.

2.3.1 SELECT-FROM-WHERE

O comando desse tipo possui a seguinte estrutura:

SELECT < atributos > **FROM** < tabelas > **WHERE** < condição >;

onde

- <atributos> são atributos cujos valores serão obtidos pela consulta;
- <tabelas> são relações para a realização da consulta;
- <condição> é uma expressão booleana, para identificar as tuplas a serem obtidas pela consulta

Exemplo baseado na figura 2.3:

Sentença: Recupere o nome do município dos casos em que o sexo é “feminino”.

Consulta em SQL: SELECT nome_munic FROM CASOS WHERE cs_sexo = “feminino”;

2.3.2 Ausência de cláusula WHERE e uso de asterisco

A cláusula **WHERE** não é obrigatória; sem ela, a consulta da subseção 2.3.1 retorna o nome do município de todos os casos:

Consulta sem cláusula WHERE: SELECT nome_munic FROM CASOS;

Para selecionar todos os atributos sem escrever todos eles na consulta, basta utilizar um asterisco (*):

Consulta que retorna todos os atributos: SELECT * FROM CASOS;

2.3.3 DISTINCT e ALL

Para eliminar tuplas duplicadas, ou seja, tuplas que aparecem mais de uma vez no resultado de uma consulta, basta utilizar a palavra chave **DISTINCT**: SELECT DISTINCT nome_munic FROM CASOS;

Na consulta "SELECT ALL nome_munic FROM CASOS;", seleciona-se todos os salários dos empregados, o que equivale a escrever "SELECT nome_munic FROM CASOS;".

2.3.4 LIKE

O LIKE é um operador de comparação de cadeia de caracteres, que pode ser usado na cláusula WHERE. Ele permite a comparação com padrões especificados usando os caracteres coringa "%" e "_".

Sentença: Recupere o nome do município dos casos em que o nome do município começa com a letra "S".

Consulta: SELECT nome_munic FROM CASOS WHERE nome_munic LIKE "S%";

Nessa consulta, "%" substitui um número arbitrário (zero ou mais) caracteres. Uma alternativa para "%" é "_", que substitui um único caractere. Por exemplo, se o nome do município solicitado precisa conter 5 letras, uma possível consulta poderia ser:

Sentença: Encontre todos os nomes de municípios com 5 letras.

Consulta: SELECT nome_munic FROM CASOS WHERE nome_munic LIKE "_ _ _ _ _";

2.3.5 ORDER BY

O **ORDER BY** é utilizado para ordenar o resultado de uma consulta por algum atributo ou conjunto de atributos. Por exemplo:

Sentença: Recupere o nome do município dos casos em que o sexo é "feminino', ordenados pelo nome do município.

Consulta: SELECT nome_munic FROM CASOS WHERE cs_sexo = "feminino" ORDER BY nome_munic;

Além disso, é possível acrescentar **ASC** ou **DESC**, para ordenar em ordem crescente ou decrescente, respectivamente. 'ASC' é a ordenação padrão nesse tipo de consulta.

Consulta com ordenação decrescente: SELECT nome_munic FROM CASOS WHERE cs_sexo ="feminino" ORDER BY nome_munic DESC;

Também é permitido colocar mais de um atributo a ser considerado na ordenação:

Consulta com ordenação envolvendo mais de um atributo: SELECT nome_munic FROM CASOS WHERE cs_sexo =“feminino” ORDER BY nome_munic DESC, cs_sexo ASC;

2.3.6 Agregações

Agregações são utilizadas para resumir informação de múltiplas tuplas em apenas uma única tupla. SQL contém algumas agregações embutidas:

- **COUNT**: retorna o número de tuplas ou valores que satisfazem a consulta. Por exemplo, a consulta a seguir retorna a quantidade de empregados em que o sexo é feminino: *SELECT COUNT(*) FROM CASOS WHERE cs_sexo =“feminino”;*
- **SUM**: retorna a soma de valores de um atributo contidos nas tuplas que satisfazem a consulta: *SELECT SUM(idade) FROM CASOS;*
- **MAX**: retorna o valor máximo de um atributo contido nas tuplas que satisfazem a consulta: *SELECT MAX(idade) FROM CASOS;*
- **MIN**: retorna o valor mínimo de um atributo contido nas tuplas que satisfazem a consulta: *SELECT MIN(idade) FROM CASOS;*
- **AVG**: retorna o valor médio de um atributo contido nas tuplas que satisfazem a consulta: *SELECT AVG(idade) FROM CASOS;*

2.3.7 GROUP BY

Utiliza-se **GROUP BY** para aplicar agregações a subgrupos de tuplas em uma relação, onde os subgrupos são baseados em valores de alguns atributos (Elmasri e Navathe, 2015).

Por exemplo, a fim de obter a idade média dos pacientes em cada município, é necessário dividir a relação em subconjuntos disjuntos de tuplas, em que cada subconjunto possui tuplas com o mesmo valor em atributos chamados **atributos de agrupamento**. Então, executa-se a agregação para cada subconjunto. “GROUP BY” especifica os atributos de agrupamento, presentes na cláusula “SELECT”, a fim de que o resultado da aplicação de cada agrupamento para um subconjunto de tuplas apareça com o valor dos atributos de agrupamento.

Sentença: Para cada município, devolva o número de pacientes no município e a idade média dos pacientes.

Consulta: SELECT nome_munic, COUNT(*), AVG(idade) FROM CASOS GROUP BY nome_munic;

A figura 2.5 mostra o resultado da consulta acima.

codigo_ibge	nome_munic	nome_drs	obito	raca_cor	idade	cs_sexo	diagnostico_covid19
3543501	São Paulo	Sorocaba	FALSE	BRANCA	32	MASCULINO	CONFIRMADO
3550308	São Paulo	Município de São Paulo	FALSE	PARDA	23	MASCULINO	CONFIRMADO
3522406	Botucatu	Sorocaba	FALSE	IGNORADO	0	FEMININO	CONFIRMADO
3507506	Botucatu	Bauru	FALSE	IGNORADO	10	MASCULINO	CONFIRMADO

nome_munic	Count(*)	Avg(Idade)
São Paulo	2	27,5
Botucatu	2	5

Figura 2.4: *Resultado da consulta com a cláusula "GROUP BY"*

Capítulo 3

Ferramentas

Neste capítulo, são mostradas as ferramentas utilizadas para o desenvolvimento deste trabalho de conclusão de curso.

3.1 NaLIR

NaLIR é uma abreviação para *Natural Language Interface to Relational Databases*, que, em Português, significa Interface de Linguagem Natural para Bancos de Dados Relacionais (Li e Jagadish, 2014). Ela é uma ferramenta de software que possui três partes principais: interpretação de consulta, comunicador interativo e tradutor de árvore de consulta.

A interpretação de consulta, que inclui o mapeador de nós da árvore de análise e o ajustador de estrutura, é responsável por interpretar uma consulta em linguagem natural e representar a interpretação como uma árvore de consulta. No comunicador interativo, há comunicação com o usuário para garantir a corretude do processo de interpretação. A árvore de consulta, possivelmente verificada pelo usuário, é traduzida em uma consulta SQL no tradutor de árvore de consulta e então executada por um SGBD relacional.

- **Analisador de dependências**: Para traduzir uma consulta em linguagem natural em uma consulta SQL, é necessário entender a consulta em linguagem natural de forma linguística. A NaLIR utiliza o *Stanford Parser* para gerar uma árvore de análise com base na consulta em linguagem natural;
- **Mapeador de nós da árvore de análise**: Identifica os nós na árvore de análise que podem ser mapeados para componentes SQL e marcados com um símbolo (*token*). No mapeamento, é possível que alguns nós não sejam mapeados em nenhum componente SQL, assim como um nó pode ter diversos mapeamentos e, nesse caso, por padrão, o mapeador retorna o melhor mapeamento para o ajustador da estrutura da árvore de análise;
- **Ajustador da estrutura da árvore de análise**: Após o mapeamento de nós, assume-se que cada nó é entendido pelo NaLIR e o próximo passo é entender corretamente a estrutura da árvore a partir da perspectiva do banco de dados. Isso não é fácil, pois a árvore de análise pode estar incorreta, fora da cobertura semântica do sistema ou

ambígua para o banco de dados. Nesses casos, ajusta-se a estrutura da árvore de análise e geram-se árvores de consulta candidatas. O ajuste é feito em duas etapas:

- **Primeira etapa**: reformulam-se os nós na árvore de análise para que eles estejam na cobertura sintática do NaLIR, obtendo-se uma árvore de análise válida. Havendo várias árvores candidatas válidas de análise para a consulta, escolhe-se a melhor para o segundo passo e informam-se as k melhores árvores para o comunicador interativo;

- **Segunda etapa**: a árvore de análise escolhida é analisada de forma semântica e nós implícitos são inseridos para tornar a árvore mais razoável semanticamente. Na implementação original do NaLIR, essa etapa é feita sob supervisão do usuário. Após a inserção de nós implícitos, obtém-se a interpretação exata para a consulta e essa representação é por meio de uma árvore de consulta.

  Árvore de consulta é uma estrutura de dados em que são representadas relações nas folhas da árvore (nós externos) e operações de álgebra relacional em nós internos.

  Grosso modo, álgebra relacional é um conjunto de operações para recuperação de dados em bancos de dados relacionais, e é a base da linguagem SQL. Uma parte dessas operações advém da teoria matemática de conjuntos (UNION, INTERSECTION, SET DIFFERENCE, CARTESIAN PRODUCT), e a outra parte consiste de operações desenvolvidas especificamente para bancos de dados relacionais (SELECT e PROJECT, por exemplo).

  Em uma árvore (A) desse tipo, executa-se a operação contida no nó interno e substitui-se o conteúdo desse nó pela relação resultante dessa operação. Inicia-se a execução a partir das folhas de A, que representam relações do banco de dados, e termina-se a execução no nó raiz, que possui todas as subárvores existentes em A.

  A figura 3.1 contém um exemplo de árvore de consulta.

- **Comunicador interativo**: o comunicador interativo explica como a consulta é processada. As comunicações interativas são organizadas em três etapas, que verificam os resultados intermediários no mapeamento de nós da árvore de análise, na reformulação da estrutura da árvore de análise e na inserção de nós implícitos. Para cada parte ambígua, gera-se um painel de seleção de múltipla escolha, no qual cada escolha corresponde a uma interpretação diferente. Toda vez que um usuário altera uma escolha, o sistema imediatamente reprocessa todas as ambiguidades nas etapas seguintes;

- **Tradutor de árvore de consulta**: O tradutor utiliza a estrutura da árvore de consulta para gerar uma estrutura adequada na expressão SQL e completa as junções de chave primária com chave estrangeira (*FK-PK join paths*, em Inglês). O comando SQL resultante pode conter funções de agregação, subconsultas multiníveis, vários tipos de junções, entre outros elementos. O NaLIR submete o comando SQL traduzido para execução no SGBD e retorna o resultado da consulta para o usuário.

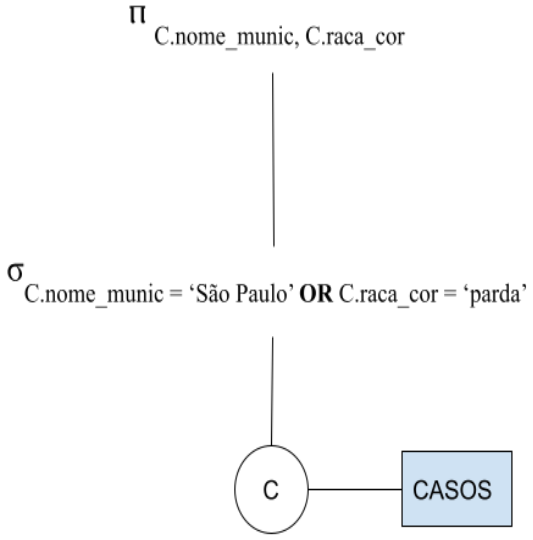


Figura 3.1: *Árvore de consulta para o comando SQL "SELECT nome_munic, raca_cor FROM casos WHERE nome_munic = "São Paulo" OR raca_cor = "parda". σ indica uma operação "SELECT" e π indica uma operação "PROJECT", em que apenas determinadas colunas de uma relação são selecionadas*

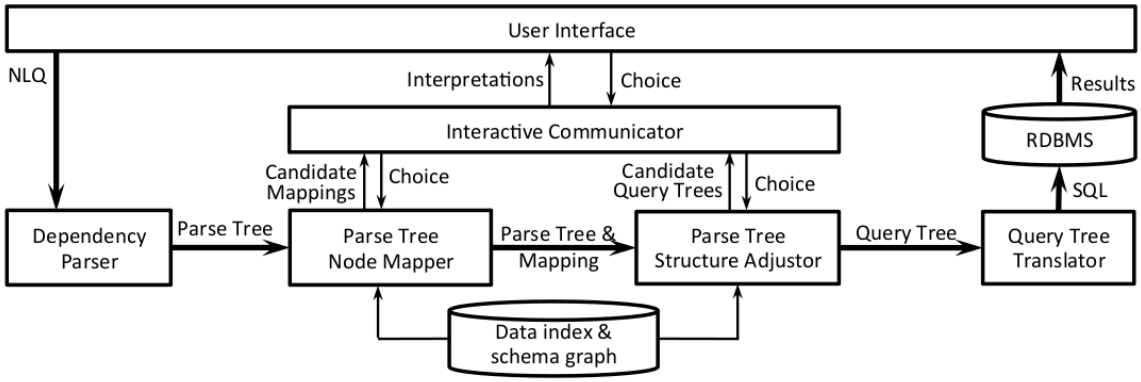


Figura 3.2: *Arquitetura do NALIR (Li e Jagadish, 2014)*

3.2 GLAMORISE

O NaLIR geralmente retorna bons resultados quando a sentença em linguagem natural não envolve agregações nem agrupamentos. Para lidar com sentenças contendo agregações e agregações, desenvolveu-se uma solução denominada GLAMORISE (Novello, 2021), cuja arquitetura é mostrada na figura 3.2. Além disso, é possível utilizar essa solução com outras implementações de interface de linguagem natural para bancos de dados diferentes do NaLIR, como, por exemplo, o Danke (Novello, 2021).

Os componentes da arquitetura do GLAMORISE são:

- **Pré-processador**: processa a consulta original em linguagem natural para remover dela (e registrar para uso posterior) os termos que façam referência a operações de agregação e agrupamento de dados. Por exemplo, na sentença "Retorne a idade

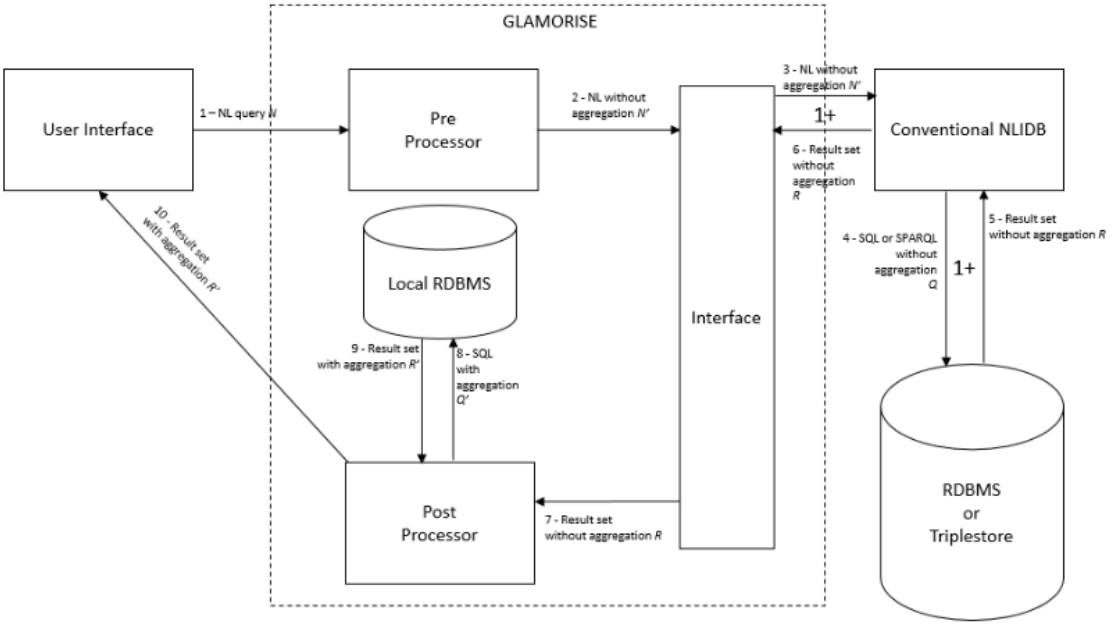


Figura 3.3: *Arquitetura do GLAMORISE (Novello, 2021)*

média em casos", o componente remove remove e registra a palavra "média";

- **Interface**: realiza a integração do GLAMORISE com uma NLIDB, que, em Inglês, significa *Natural Language Interface to Database* e, em tradução livre para o Português, significa Interface de Linguagem Natural para Banco de Dados. Nesse componente, a consulta pré-processada (já sem os termos de agregação e agrupamento) é passada para a NLIDB para ser traduzida e executada, e que os dados devolvidos como resposta da execução da consulta são gravados em um BD temporário;
- **Pós-processador**: componente no qual a consulta SQL é construída com base nos metadados salvos no pré-processador. Além disso, a consulta SQL construída é executada no BD temporário gerado pelo componente anterior e inclui as agregações e agrupamentos removidos pelo pré-processador.

O GLAMORISE é capaz de tratar melhor as agregações e agrupamentos, pois ele associa as agregações e os agrupamentos com os atributos, seguindo os padrões contidos no arquivo "glamorise_nalir_adapted.json".

Capítulo 4

Desenvolvimento do trabalho: Adaptações

Para realizar esta monografia, foi utilizada a versão 3.10 da linguagem Python. O *link* dos repositórios originais é:

- GLAMORISE: https://github.com/novello/GLAMORISE
- NaLIR: https://github.com/novello/nalir-glamorise

O *link* para os sistemas adaptados é https://github.com/anttoniol/tcc_sistemas.

Neste trabalho, adaptaram-se as ferramentas de software NaLIR e GLAMORISE para que possam ser usadas com consultas e bancos de dados em língua portuguesa. Nas seções a seguir, descrevem-se as modificações feitas nos códigos-fonte das duas ferramentas.

Os arquivos de configuração são utilizados para fornecer dados e/ou instruções ao NaLIR e ao GLAMORISE.

4.1 Inclusão de arquivos de configuração

- **glamorise_nalir_adapted.json**: informa a classificação morfológica usual de determinadas palavras em frases escritas em Português, assim como correspondências entre alguns comandos SQL e palavras em Português. O conteúdo desse arquivo é utilizado nos componentes "Pré-processador" e "Interface" do GLAMORISE, e um trecho desse arquivo está na figura 4.1;
- **nalir_tokens_adapted.xml**: categoriza determinados termos em Português que devem ser substituídos por comandos SQL. O conteúdo desse arquivo é utilizado no componente "Interface" do GLAMORISE e um trecho desse arquivo está na figura 4.2;
- **tokens_adapted.xml**: mesma utilização do item anterior, mas no NaLIR, onde o conteúdo desse arquivo é utilizado no componente "Mapeador de Nós da Árvore de Análise".

```
33      "default_pattern": [
34          {
35              "POS": {"IN": ["NOUN", "PROPN"]},
36              "LOWER": {
37                  "NOT_IN": [
38                      "número"
39                  ]
40              }
41          },
42          {
43              "POS": "ADJ",
44              "OP": "*"
45          }
46      ],
```

Figura 4.1: *Trecho de código do arquivo "glamorise_nalir_adapted.json". Nesse trecho, informa-se um padrão em Português em que um substantivo comum ou próprio geralmente é sucedido por um ou nenhum adjetivo*

```
5   <!-- Command Token, verb -->
6   <CMT_V>
7   <phrase>qual
8   <example>Qual foi a produção de óleo no estado do Rio de Janeiro?</example>
9   </phrase>
10  <phrase>quais
11  <example>Quais campos produzem mais óleo por mês?</example>
12  </phrase>
13  <phrase>quantos
14  <example>Quantos campos há no Paraná?</example>
15  </phrase>
```

Figura 4.2: *Trecho de código do arquivo "nalir_tokens_adapted.xml". Nesse trecho, informa-se que determinados termos ("qual", "quais", "quantos", etc.) são verbos classificados como tokens de comando (command token)*

Nos dois primeiros itens acima, o uso da palavra "nalir" no nome dos arquivos indica que, no GLAMORISE, o NLIDB considerado nesses arquivos é o NaLIR. No último item, não há a palavra "nalir", pois, no repositório que contém os sistemas adaptados, esse arquivo já está no diretório do NaLIR.

4.2 Alteração de arquivos

- **Substituição de termos em Inglês por termos em Português**: Alguns dos arquivos de código fonte das ferramentas contêm strings constantes com termos em Inglês (que são procurados na consulta em linguagem natural para serem mapeados para SQL). Esses termos precisaram ser substituídos pelos termos em Português correspondentes. Os arquivos afetados por tais modificações foram os seguintes:
  - glamorise.py: contém a classe "Glamorise", que remove os termos referentes a agrupamento e agregação, para, mais tarde, utilizá-los no conjunto de dados retornado pelo NaLIR. Esse arquivo pertence ao diretório do GLAMORISE e um trecho desse arquivo está na figura 4.5;

```
716
717         # build the final query
718         self.__pos_glamorise_sql = (self.__select_clause + self.__from_clause + self.__group_by_clause + ' ' + \
719                                     self.__having_clause + ' ' + self.__order_by_clause).replace("  ", " ").strip() + \
720                                    " LIMIT " + str(limit)
721
```

Figura 4.3: *Trecho de código do arquivo "glamorise.py". Nesse trecho, ocorre a formação do comando SQL final. Para isso, reúnem-se os elementos identificados como parte das cláusulas "SELECT", "FROM", "GROUP BY", "HAVING" e "ORDER BY". Não necessariamente todas essas cláusulas aparecem no comando SQL final*

- glamorise_nlidb.py: possui a classe "GlamoriseNlidb", que é filha da classe "Glamorise", considerando o conceito de herança em programação orientada a objeto. Nela, identifica-se, por exemplo, o NLIBD a ser utilizado, além de estabelecer uma interface entre o GLAMORISE e o NLIDB. Esse arquivo pertence ao diretório do GLAMORISE;
- explainer.py: responsável pela inserção de nós na árvore de análise a fim de facilitar a tradução da consulta em Português para SQL. Esse arquivo pertence ao diretório do NaLIR, no qual o conteúdo desse arquivo é utilizado entre os componentes "Ajustador da Estrutura da Árvore de Análise" e "Tradutor da Árvore de Consulta";
- node_mapper.py: possui o papel de classificar os nós da árvore de análise, com base nos tokens contidos no arquivo "tokens_adapted.xml". Além disso, o algoritmo remove nós considerados inúteis, além de identificar os nós com mais chances de corresponder a atributos e valores do banco de dados. Esse arquivo pertence ao diretório do NaLIR, no qual o conteúdo desse arquivo é utilizado no componente "Mapeador de Nós da Árvore de Análise";

```
24      def phrase_process(query, db, config, tokens = ''):
25          if tokens == '':
26              tokens = f'{config.zfiles_path}/tokens_adapted.xml'
27              tokens = parse(tokens)
28
29          NodeMapper.tokenizer(query, tokens)
30          NodeMapper.delete_useless(query)
31          NodeMapper.map(query,db)
32          NodeMapper.individual_rank(query)
33          group_ranking = NodeMapper.group_ranking(query,db)
34          return NodeMapper.group_ranking_to_df(group_ranking)
```

Figura 4.4: *Trecho de código do arquivo "node_mapper.py". Nesse trecho, os nós da árvore de análise são categorizados, alguns nós considerados inúteis são removidos e estabelecem-se correspondências entre nós e atributos/valores contidos no banco de dados*

- stanford_parser.py: tem o papel de criar a árvore de análise, utilizando funções e classes auxiliares. No projeto original, utiliza a biblioteca *Stanford Parser* para formar a árvore de análise. Esse arquivo pertence ao diretório do NaLIR,

no qual o conteúdo desse arquivo é utilizado no componente "Analisador de dependência", e um trecho desse arquivo está na figura 4.6;

```
32
33          self.parse_adapted(query)
34
35          self.build_tree(query)
36          self.fix_conj(query)
37
38          self.print_tree_table(query)
39
40          print("Parse Tree:\n")
41          print(query.parse_tree)
42          print("")
```

Figura 4.5: *Trecho de código do arquivo "stanford_parser.py". Nesse trecho, utilizam-se funções auxiliares para a construção de uma tabela auxiliar para a formação da árvore de análise*

  - similarity.py: contém um algoritmo que busca a melhor similaridade entre nós da árvore de análise e atributos/valores no banco de dados. Esse arquivo pertence ao diretório do NaLIR, no qual o conteúdo desse arquivo é utilizado no componente "Mapeador de Nós da Árvore de Análise".

- **Alteração na categorização de palavras, utilizada na tokenização**:
  - node_mapper.py: descrito em itens anteriores.
- **Utilização da biblioteca Spacy, com o modelo pt_core_news_md**:

  O código original do NaLIR usa a biblioteca NLTK para fazer o processamento das consultas em linguagem natural em Inglês. Apesar de ser capaz de processar textos em várias linguagens, a NLTK tem limitações para o processamento de frases em Português, o que motivou a pesquisa por outras bibliotecas e resultou na escolha do Spacy como substituta.

  A Spacy é uma biblioteca disponível na linguagem de programação Python, para processamento de linguagem natural em mais de 60 idiomas. Com essa biblioteca, é possível obter diferentes classificações de palavras em uma frase, assim como a relação que essas palavras possuem com as outras da mesma frase (Explosion, 2022a).

  A substituição do uso da NLTK pelo uso da Spacy se deu nos seguintes arquivos já descritos anteriormente:

  - glamorise.py;
  - glamorise_nlidb.py;
  - node_mapper.py;
  - stanford_parser.py;
  - similarity.py.

- **Utilização da biblioteca NLTK em Português**: assim como Spacy, o NLTK é uma biblioteca utilizada para processamento de linguagem natural em Python e também

está disponível em muitas linguagens. Entretanto, quando comparada com o Spacy, o NLTK tem mais limitações para processamento de frases em Português, o que motivou a pesquisa por outras bibliotecas e resultou na escolha do Spacy como substituta (Project, 2022).

- **Outras alterações**:
  - **stanford_parser.py**: não há uma versão da biblioteca *Stanford Parser* em Português, assim foi necessária a implementação de um algoritmo que constrói a lista de dependências, que é utilizada na construção na tabela de árvore, que, por sua vez, é usada na construção da árvore de *parse*.
  - **similarity.py**: uso da biblioteca Spacy para obter o lema de cada palavra em frases escritas em Português;

4.3 Uso geral do NaLIR e do GLAMORISE

Nesta monografia, utilizou-se o GLAMORISE apenas com o NaLIR, embora a solução proposta por Novello, 2021 possa ser utilizada com outros sistemas diferentes do NaLIR. Outras mudanças no uso dos dois sistemas estão a seguir:

- **Ajustador da estrutura da árvore de análise no NaLIR**: na segunda etapa do ajuste na estrutura, não foi realizada a supervisão do usuário;
- **Comunicador interativo do NaLIR**: não foi utilizado nos experimentos desta monografia.

Capítulo 5

Experimentos e resultados

Para verificar o funcionamento e o desempenho das ferramentas após as adaptações realizadas, especificou-se um conjunto de consultas em português sobre um banco de dados real que está publicamente disponível na internet. As consultas envolvem uma grande variedade de operações de filtragem, agrupamento e agregação de dados.

As seções a seguir descrevem o banco de dados usado, algumas consultas selecionadas e os resultados obtidos do seu mapeamento para SQL usando as ferramentas adaptadas para a língua portuguesa.

5.1 Dados

Utilizaram-se dados disponibilizados pela Fundação SEADE, referentes a casos de COVID-19 no Brasil entre 2020 e 2022. Durante o desenvolvimento desta monografia, esses dados foram constantemente atualizados pela Fundação SEADE em um repositório no Github (SEADE, 2022a). Para este trabalho, escolheu-se uma das versões de um conjunto vasto de dados sobre casos de COVID-19 por raça/cor e município (SEADE, 2022b).

5.1.1 Estrutura

No repositório da Fundação SEADE, os dados usados neste trabalho estavam em um arquivo com extensão csv, cujas colunas com as respectivas descrições estão na tabela 5.1.

Variável	Descrição
codigo_ibge	Código do município no IBGE (7 dígitos) de residência do paciente
nome_munic	Nome do município de residência do paciente
nome_drs	Nome do Dpto. Regional de Saúde de residência do paciente
obito	Indica se o paciente veio a óbito por COVID-19
raca_cor	Cor ou raça do paciente

idade	Idade do paciente
cs_sexo	Sexo do paciente
diagnostico_covid19	Confirmação de COVID-19
asma	Paciente apresenta esse fator de risco (asma)
cardiopatia	Paciente apresenta esse fator de risco (cardiopatia)
diabetes	Paciente apresenta esse fator de risco (diabetes)
doenca_hematologica	Paciente apresenta esse fator de risco (doença hematológica)
doenca_hepatica	Paciente apresenta esse fator de risco (doença hepática)
doenca_neurologica	Paciente apresenta esse fator de risco (doença neurológica)
doenca_renal	Paciente apresenta esse fator de risco (doença renal)
imunodepressao	Paciente apresenta esse fator de risco (imunodepressão)
obesidade	Paciente apresenta esse fator de risco (obesidade)
pneumopatia	Paciente apresenta esse fator de risco (pneumopatia)
puerpera	Paciente se encontra nesse estágio (puérpera)
sindrome_de_down	Paciente apresenta esse fator de risco (síndrome de down)

Tabela 5.1: *Dicionário para os atributos da relação "CASOS"*

5.1.2 Banco de dados

O banco de dados denominado "seade" foi implementado no MySQL, o qual é um sistema de gerenciamento de banco de dados relacionais de código aberto e que tem suporte à linguagem SQL (Oracle, 2022). O banco "seade" contém uma tabela chamada "casos", cujas colunas são as mesmas do arquivo csv mencionado na subseção 5.1.1. Como o arquivo era muito grande (cerca de 5 milhões de linhas), decidiu-se armazenar 5000 registros aleatórios do csv no banco de dados. A fim de obter os registros aleatórios, implementou-se um algoritmo em Python para a leitura do csv e para a seleção randômica dos registros.

5.2 Arquivo de configuração para um BD específico

glamorise_interface_nalir_<BD>.json: utilizado para mapear atributos da(s) tabela(s) do banco "<BD>" a sinônimos que podem aparecer nas consultas em linguagem natural. Esse arquivo precisa ser configurado especificamente para o BD que será consultado. O conteúdo desse arquivo é utilizado no componente "Interface" do GLAMORISE e um trecho desse arquivo está na figura 5.1;

```
    "código": "casos.codigo_ibge",
    "município": "casos.nome_munic",
    "região": "casos.regiao",
    "óbito": "casos.obito",
    "raça": "casos.raca_cor",
    "idade": "casos.idade",
    "sexo": "casos.cs_sexo",
    "diagnóstico": "casos.diagnostico_covid19",
    "asma": "casos.asma",
    "cardiopatia": "casos.cardiopatia",
    "diabetes": "casos.diabetes",
    "doença_hematológica": "casos.doenca_hematologica",
    "doença_hepática": "casos.doenca_hepatica",
    "doença_neurológica": "casos.doenca_neurologica",
    "doença_renal": "casos.doenca_renal",
    "imunodepressão": "casos.imunodepressao",
    "obesidade": "casos.obesidade",
    "pneumopatia": "casos.pneumopatia",
    "puérpera": "casos.puerpera",
    "síndrome_de_down": "casos.sindrome_de_down"
```

Figura 5.1: *Trecho de código do arquivo "glamorise_interface_nalir_seade.json". Nesse trecho, mostra-se a correspondência entre palavras (ou termos compostos separados por um "_") em Português e os atributos da tabela "casos" no banco "seade"*

5.3 Arquivos para teste dos sistemas adaptados

- **main_nalir_seade_single_nlq.py**: utilizado para testar o sistema com sentenças escritas em Português, próprias para o banco "seade". O conteúdo desse arquivo faz parte do componente "Interface do Usuário", mostrado na figura 3.3, e um trecho desse arquivo está na figura 5.2;

- **main_nalir_seade_single_nlq_notebook.py**: parecido com o arquivo do item anterior, com a diferença de ser utilizado junto com um arquivo no formato .ipynb, para testar o GLAMORISE e o NaLIR. O conteúdo desse arquivo faz parte do componente "Interface do Usuário", mostrado na figura 3.3.

```
size = len(nlqs)
for i in range(size):
    # create GLAMORISE object (the child class is instantiated)
    glamorise = GlamoriseNlidb(NLIDB='NaLIR', config_glamorise_param=config_glamorise,
                               config_glamorise_interface_param=config_glamorise_interface, config_db=config_db,
                               tokens=nalir_tokens)
    file = open("outputs/nlqs" + str(i) + extension, "w")
    nlq_str, result, dep, ent = mc.print_results(glamorise, nlqs[i], skip)
    result2 = mc.print_total_timers()

    if extension == ".html":
        file.write('SENTENÇA: ' + nlqs[i] + '</br></br>' + result + glamorise.pd.to_html() + result2)
    elif extension == ".txt":
        file.write('SENTENÇA: ' + nlqs[i] + '\n\n' + result + glamorise.pd.to_string() + result2)

    file.close()
    del glamorise
```

Figura 5.2: *Trecho de código do arquivo "main_nalir_seade_single_nlq.py". Nesse trecho, um objeto da classe "GlamoriseNlidb" é criado e, na função da linha 75, esse objeto é utilizado para obter o comando SQL de uma sentença em Português, localizada na posição "i" da lista "nlqs". Certas partes do resultado desse processamento foram salvas em arquivos utilizados durante os experimentos*

5.4 Consultas

Para cada consulta das próximas subseções, há uma figura com o diagrama da relação de dependência das palavras da consulta e, em seguida, uma outra figura com o resultado do processamento dessa consulta no NaLIR e no GLAMORISE. Para algumas consultas, o diagrama e o resultado do processamento estão nesta seção; para outras, o diagrama e o resultado estão nos apêndices. No resultado, há as seguintes informações:

- **Natural Language Query**: consulta em Português processada no NaLIR e no Glamorise;
- **Tree Table**: uma lista em que cada item possui o formato *[index, word, pos, parent_index, parent_word, dep]*, onde:
  - **index**: índice dado ao elemento *word* no processamento da consulta;
  - **word**: palavra que faz parte da consulta em Português;
  - **pos**: classe gramatical da palavra na consulta (*coarse-grained part-of-speech*, ou POS), seguindo o padrão descrito em Contributors, 2022b;
  - **parent_index**: índice de uma palavra com a qual *word* tem uma relação de dependência;
  - **parent_word**: palavra cujo índice é *parent_index*;
  - **dep**: relação de dependência sintática entre *word* e *parent_word* na consulta, seguindo o padrão descrito em Contributors, 2022a
- **Parse Tree**: representação bastante simples da dependência entre as palavras da consulta. Nessa representação, um nó x tem dependência com um nó y se y é pai de x nessa árvore;
- **pre_prepared_query**: consulta pré-processada pelo GLAMORISE, para a identificação de agrupamentos e de agregações, que não são passados para o NaLIR;
- **group_ranking_to_df**: palavras da consulta mapeadas para os atributos da tabela "casos";
- **nlidb_interface_sql**: comando SQL que o NaLIR retorna para o GLAMORISE;
- **pos_glamorise_sql**: comando SQL executado nos registros obtidos da execução de *nlidb_interface_sql*;
- **Results**: resposta final para a consulta.

5.4.1 Sem agregações e sem cláusula "WHERE"

Retorne todos os municípios em casos: ver figuras A.1 e A.2.

5.4.2 Sem agregações e com cláusula “WHERE”

- **Retorne todas as idades dos casos onde o município é São Paulo**: ver figuras 5.3 e 5.4.;
- **Retorne todas as idades dos casos do município de São Paulo**: ver figuras 5.5 e 5.6.

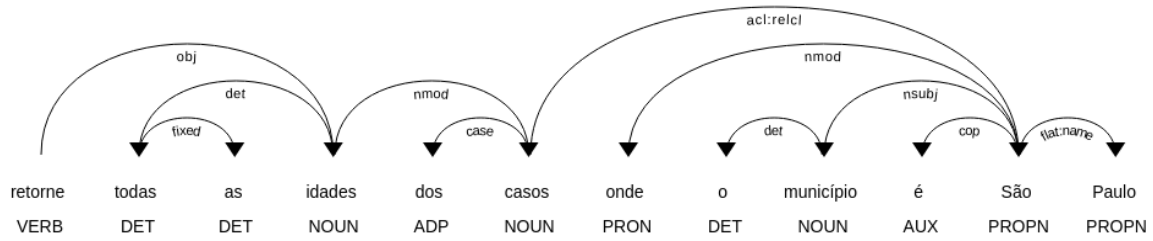


Figura 5.3: *Diagrama de dependências da consulta "Retorne todas as idades dos casos onde o município é São Paulo", mencionada na seção 5.4.2*

Natural Language Query: retorne todas as idades dos casos onde o município é São Paulo**

Tree Table:

[1, 'retorne', 'e', 0, 'ROOT', 'root']
[2, 'todas', 'DET', 4, 'idades', 'det']
[3, 'as', 'DET', 2, 'todas', 'fixed']
[4, 'idades', 'NOUN', 1, 'retorne', 'obj']
[5, 'dos', 'ADP', 6, 'casos', 'case']
[6, 'casos', 'NOUN', 4, 'idades', 'nmod']
[7, 'onde', 'PRON', 11, 'São', 'nmod']
[8, 'o', 'DET', 9, 'município', 'det']
[9, 'município', 'NOUN', 11, 'São', 'nsubj']
[10, 'é', 'AUX', 11, 'São', 'cop']
[11, 'São', 'PROPN', 6, 'casos', 'acl:relcl']
[12, 'Paulo', 'PROPN', 11, 'São', 'flat:name']

Parse Tree:

(31)ROOT
  (32)retorne
    (33)idades
      (34)todas
        (35)as
      (36)casos
        (37)dos
        (38)São
          (39)onde
          (40)município
            (41)o
          (42)é
          (43)Paulo

GLAMORISE Internal Properties

Preprocessor Properties
pre_prepared_query = 'retorne todas as idades dos casos onde o município é São Paulo'

NLIDB Interface Properties

group_ranking_to_df:
idades:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties

pos_glamorise_sql = 'SELECT * FROM NLIDB_result_set LIMIT 100'

pos_group_by_fields = ['casos_idade']

Results

	casos_idade
0	41.0
1	54.0
2	22.0
3	51.0
4	54.0
..	...
95	28.0
96	28.0
97	43.0
98	27.0
99	21.0

[100 rows x 1 columns]

Figura 5.4: *Resultado simplificado do processamento da consulta "Retorne todas as idades dos casos onde o município é São Paulo", mencionada na seção 5.4.2*

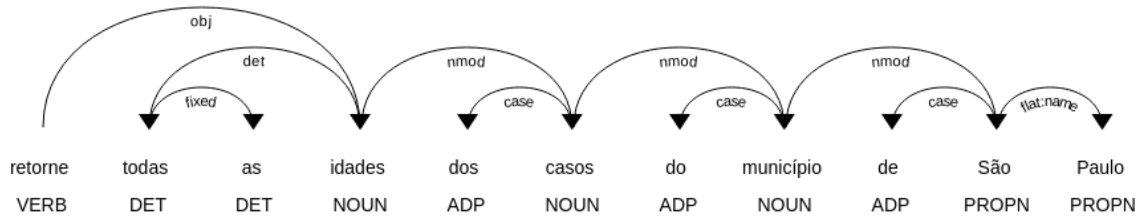


Figura 5.5: *Diagrama de dependências da consulta "Retorne todas as idades dos casos do município de São Paulo", mencionada na seção 5.4.2*

Natural Language Query: retorne todas as idades dos casos do município de São Paulo**

Tree Table:

[1, 'retorne', 'c', 0, 'ROOT', 'root']
[2, 'todas', 'DET', 4, 'idades', 'det']
[3, 'as', 'DET', 2, 'todas', 'fixed']
[4, 'idades', 'NOUN', 1, 'retorne', 'obj']
[5, 'dos', 'ADP', 6, 'casos', 'case']
[6, 'casos', 'NOUN', 4, 'idades', 'nmod']
[7, 'do', 'ADP', 8, 'município', 'case']
[8, 'município', 'NOUN', 6, 'casos', 'nmod']
[9, 'de', 'ADP', 10, 'São', 'case']
[10, 'São', 'PROPN', 8, 'município', 'nmod']
[11, 'Paulo', 'PROPN', 10, 'São', 'flat:name']

Parse Tree:

(45)ROOT
  (46)retorne
    (47)idades
      (48)todas
        (49)as
      (50)casos
        (51)dos
        (52)município
          (53)do
          (54)São
            (55)de
            (56)Paulo

GLAMORISE Internal Properties

Preprocessor Properties
pre_prepared_query = 'retorne todas as idades dos casos do município de São Paulo'

NLIDB Interface Properties

group_ranking_to_df:
idades:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties

pos_glamorise_sql = 'SELECT * FROM NLIDB_result_set LIMIT 100'

pos_group_by_fields = ['casos_idade']

Results

	casos_idade
0	41.0
1	54.0
2	22.0
3	51.0
4	54.0
..	...
95	28.0
96	28.0
97	43.0
98	27.0
99	21.0

[100 rows x 1 columns]

Figura 5.6: *Resultado simplificado do processamento da consulta "Retorne todas as idades dos casos do município de São Paulo", mencionada na seção 5.4.2*

5.4.3 Agregações sem agrupamentos e sem cláusula “WHERE”

- Usando COUNT:

  Devolva quantos óbitos existem em casos: ver figuras 5.7 e 5.8.

  Devolva quantos casos há: ver figuras 5.9 e 5.10.

- Usando MAX:

  Devolva a máxima idade em casos: ver figuras A.3 e A.4.

- Usando MIN:

  Devolva a mínima idade em casos: ver figuras A.5 e A.6.

- Usando SUM:

  Devolva as idades somadas em casos: ver figuras A.7 e A.8.

- Usando AVG:

  Devolva a idade média em casos: ver figuras A.9 e A.10.

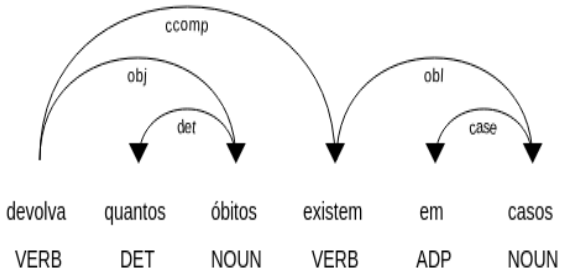


Figura 5.7: *Diagrama de dependências da consulta "Devolva quantos óbitos existem em casos", mencionada na seção 5.4.3*

Natural Language Query: devolva quantos óbitos existem em casos**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'óbitos', 'NOUN', 1, 'devolva', 'obj']
[3, 'existem', 'VERB', 1, 'devolva', 'parataxis']
[4, 'em', 'ADP', 5, 'casos', 'case']
[5, 'casos', 'NOUN', 3, 'existem', 'obl']

Parse Tree:

(12)ROOT
  (13)devolva
    (14)óbitos
    (15)existem
      (16)casos
        (17)em

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_obito']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos']
pre_prepared_query = 'devolva óbitos existem em casos'

NLIDB Interface Properties

group_ranking_to_df: óbitos:casos.obito:NT;

nlidb_interface_sql = 'SELECT casos.obito as casos_obito FROM casos'

Postprocessor Properties
pos_aggregation_fields = ['casos_obito']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT count(casos_obito) as count_casos_obito FROM NLIDB_result_set LIMIT 100'

Results
  count_casos_obito
0 5000

Figura 5.8: *Resultado simplificado do processamento da consulta "Devolva quantos óbitos existem em casos", mencionada na seção 5.4.3*

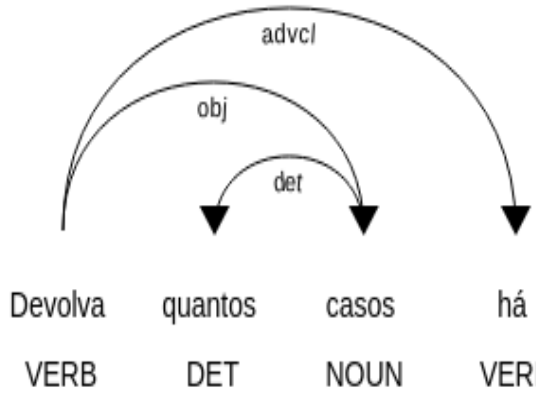


Figura 5.9: *Diagrama de dependências da consulta "Devolva quantos casos há", mencionada na seção 5.4.3*

Natural Language Query: Devolva quantos casos há

Tree Table:

[1, 'Devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'casos', 'NOUN', 1, 'Devolva', 'obj']
[3, 'há', 'VERB', 1, 'Devolva', 'advcl']

Parse Tree:

(24)ROOT
  (25)Devolva
    (26)casos
    (27)há

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_diagnostico_covid19']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos']
pre_prepared_query = 'Devolva casos há'

NLIDB Interface Properties

group_ranking_to_df: casos:casos.cs_sexo:NT;

nlidb_interface_sql = 'SELECT casos.cs_sexo as casos_cs_sexo, casos.diagnostico_covid19 as casos_diagnostico_covid19 FROM casos'

Postprocessor Properties
pos_aggregation_fields = ['casos_diagnostico_covid19']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT casos_cs_sexo, count(casos_diagnostico_covid19) as count_casos_diagnostico_covid19 FROM NLIDB_result_set GROUP BY casos_cs_sexo ORDER BY casos_cs_sexo LIMIT 100'

pos_group_by_fields = ['casos_cs_sexo']

Results

	casos_cs_sexo	count_casos_diagnostico_covid19
0	FEMININO	2747
1	INDEFINIDO	4
2	MASCULINO	2249

Figura 5.10: *Resultado simplificado do processamento da consulta "Devolva quantos casos há", mencionada na seção 5.4.3*

5.4.4 Agregações sem agrupamentos e com cláusula “WHERE”

- Usando COUNT:
  - **Devolva quantos óbitos do tipo positivo existem em casos**: ver figuras A.11 e A.12.
  - **Devolva quantos óbitos do tipo positivo onde o município é São Paulo**: ver figuras 5.11 e 5.12.
  - **Devolva quantos óbitos do tipo positivo há no município de São Paulo**: ver figuras 5.13 e 5.14.
- Usando MAX:

  Devolva a máxima idade no município de São Paulo: ver figuras A.13 e A.14.
- Usando MIN:

  Devolva a mínima idade no município de São Paulo: ver figuras A.15 e A.16.
- Usando SUM:

  Devolva as idades somadas no município de São Paulo: ver figuras A.17 e A.18.
- Usando AVG:

  Devolva a idade média no município de São Paulo: ver figuras A.19 e A.20.

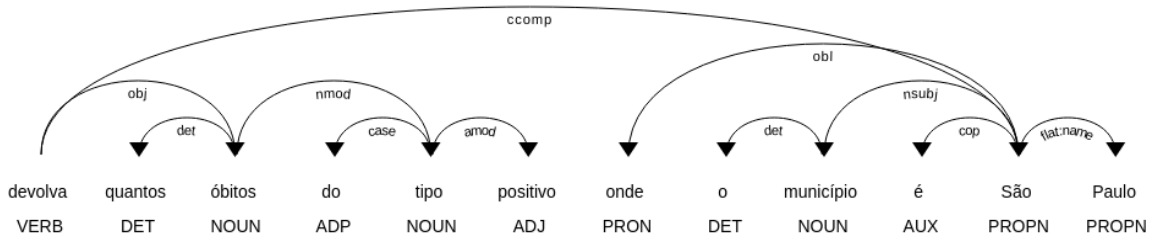


Figura 5.11: *Diagrama de dependências da consulta "Devolva quantos óbitos do tipo positivo onde o município é São Paulo", mencionada na seção 5.4.4*

****Natural Language Query**: devolva quantos óbitos do tipo positivo onde o município é São Paulo**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'óbitos', 'NOUN', 1, 'devolva', 'obj']
[3, 'do', 'ADP', 4, 'tipo', 'case']
[4, 'tipo', 'NOUN', 2, 'óbitos', 'nmod']
[5, 'positivo', 'ADJ', 4, 'tipo', 'amod']
[6, 'onde', 'PRON', 10, 'São', 'nmod']
[7, 'o', 'DET', 8, 'município', 'det']
[8, 'município', 'NOUN', 10, 'São', 'nsubj']
[9, 'é', 'AUX', 10, 'São', 'cop']
[10, 'São', 'PROPN', 2, 'óbitos', 'acl:relcl']
[11, 'Paulo', 'PROPN', 10, 'São', 'flat:name']

Parse Tree:

(29)ROOT
(30)devolva
(31)óbitos
(32)tipo
(33)do
(34)positivo
(35)São
(36)onde
(37)município
(38)o
(39)é
(40)Paulo

****GLAMORISE Internal Properties****

****Preprocessor Properties****
pre_aggregation_fields = ['casos_obito']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos']
pre_prepared_query = 'devolva óbitos do tipo positivo onde o município é São Paulo'

****NLIDB Interface Properties****

group_ranking_to_df: óbitos:casos.obito:NT;positivo:casos.obito:VT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.obito as casos_obito FROM casos WHERE casos.obito LIKE "%positivo%" AND casos.nome_munic LIKE "%Paulo%"'

****Postprocessor Properties****
pos_aggregation_fields = ['casos_obito']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT count(casos_obito) as count_casos_obito FROM NLIDB_result_set LIMIT 100'

****Results****
count_casos_obito
0 43

Figura 5.12: *Resultado simplificado do processamento da consulta "Devolva quantos óbitos do tipo positivo onde o município é São Paulo", mencionada na seção 5.4.4*

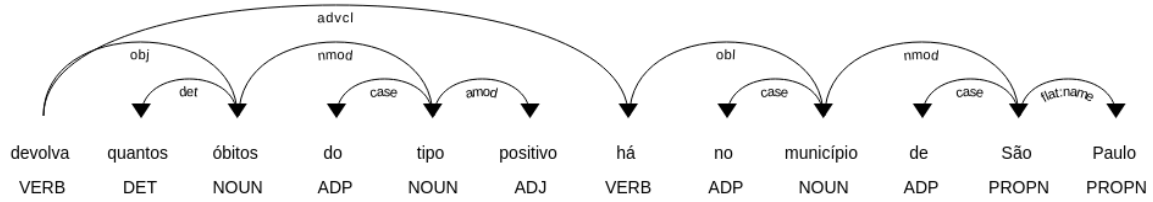


Figura 5.13: *Diagrama de dependências da consulta "Devolva quantos óbitos do tipo positivo há no município de São Paulo", mencionada na seção 5.4.4*

****Natural Language Query**: devolva quantos óbitos do tipo positivo há no município de São Paulo**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'óbitos', 'NOUN', 1, 'devolva', 'obj']
[3, 'do', 'ADP', 4, 'tipo', 'case']
[4, 'tipo', 'NOUN', 2, 'óbitos', 'nmod']
[5, 'positivo', 'ADJ', 4, 'tipo', 'amod']
[6, 'há', 'VERB', 1, 'devolva', 'advcl']
[7, 'no', 'ADP', 8, 'município', 'case']
[8, 'município', 'NOUN', 6, 'há', 'obl']
[9, 'de', 'ADP', 10, 'São', 'case']
[10, 'São', 'PROPN', 8, 'município', 'nmod']
[11, 'Paulo', 'PROPN', 10, 'São', 'flat:name']

Parse Tree:

(42)ROOT
(43)devolva
(44)óbitos
(45)tipo
(46)do
(47)positivo
(48)há
(49)município
(50)no
(51)São
(52)de
(53)Paulo

****GLAMORISE Internal Properties****

****Preprocessor Properties****
pre_aggregation_fields = ['casos_obito']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos']
pre_prepared_query = 'devolva óbitos do tipo positivo há no município de São Paulo'

****NLIDB Interface Properties****

group_ranking_to_df:
óbitos:casos.obito:NT;positivo:casos.obito:VT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.obito as casos_obito FROM casos WHERE casos.obito LIKE "%positivo%" AND casos.nome_munic LIKE "%Paulo%"'

****Postprocessor Properties****
pos_aggregation_fields = ['casos_obito']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT count(casos_obito) as count_casos_obito FROM NLIDB_result_set LIMIT 100'

****Results****
count_casos_obito
0 43

Figura 5.14: *Resultado simplificado do processamento da consulta "Devolva quantos óbitos do tipo positivo há no município de São Paulo", mencionada na seção 5.4.4*

5.4.5 Agregações com agrupamentos e sem cláusula “WHERE”

- Usando COUNT:

  Devolva quantos casos há por município: ver figuras 5.15 e 5.16.

- Usando MAX:

  Devolva a máxima idade por município: ver figuras A.21 e A.22.

- Usando MIN:

  Devolva a mínima idade por município: ver figuras A.23 e A.24.

- Usando SUM:

  Devolva as idades somadas por município: ver figuras A.25 e A.26.

- Usando AVG:

  Devolva a idade média por município: ver figuras A.27 e A.28.

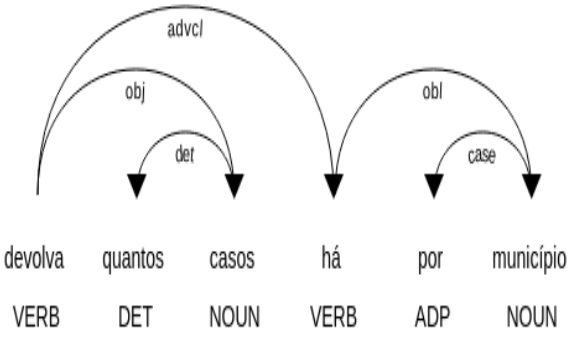


Figura 5.15: *Diagrama de dependências da consulta "Devolva quantos casos há por município", mencionada na seção 5.4.5*

Natural Language Query: devolva quantos casos há por município

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'casos', 'NOUN', 1, 'devolva', 'obj']
[3, 'há', 'ADP', 1, 'devolva', 'advcl']
[4, 'município', 'NOUN', 3, 'há', 'obj']

Parse Tree:

(104)ROOT
  (105)devolva
    (106)casos
    (107)há
      (108)município

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_diagnostico_covid19']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_nome_munic']
pre_prepared_query = 'devolva casos há município'

NLIDB Interface Properties

group_ranking_to_df:
casos:casos.cs_sexo:NT;município:casos.nome_munic:NT;

nlidb_interface_sql = 'SELECT casos.cs_sexo as casos_cs_sexo, casos.diagnostico_covid19 as casos_diagnostico_covid19, casos.nome_munic as casos_nome_munic FROM casos'

Postprocessor Properties
pos_aggregation_fields = ['casos_diagnostico_covid19']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT casos_nome_munic, casos_cs_sexo, count(casos_diagnostico_covid19) as count_casos_diagnostico_covid19 FROM NLIDB_result_set GROUP BY casos_nome_munic, casos_cs_sexo ORDER BY casos_nome_munic, casos_cs_sexo LIMIT 100'

pos_group_by_fields = ['casos_cs_sexo']

Results

	casos_nome_munic	casos_cs_sexo	count_casos_diagnostico_covid19
0	Adamantina	FEMININO	3
1	Adamantina	MASCULINO	3
2	Aguaí	FEMININO	2
3	Aguaí	MASCULINO	6
4	Agudos	FEMININO	2
..	...	...	..
95	Brotas	MASCULINO	3
96	Buritama	FEMININO	2
97	Buritama	MASCULINO	3
98	Cabreúva	FEMININO	7
99	Cabreúva	MASCULINO	4

[100 rows x 3 columns]

Figura 5.16: *Resultado simplificado do processamento da consulta "Devolva quantos casos há por município", mencionada na seção 5.4.5*

5.4.6 Agregações com agrupamentos e com cláusula “WHERE”

Por raça/cor

- Usando COUNT:

  Devolva quantos óbitos do tipo positivo há no município de São Paulo por raça: ver figuras 5.17 e 5.18.

- Usando MAX:

  Devolva a máxima idade no município de São Paulo por raça: ver figuras A.29 e A.30.

- Usando MIN:

  Devolva a mínima idade no município de São Paulo por raça: ver figuras A.31 e A.32.

- Usando SUM:

  Devolva as idades somadas no município de São Paulo por raça: ver figuras A.33 e A.34.

- Usando AVG:

  Devolva a idade média no município de São Paulo por raça: ver figuras A.35 e A.36.

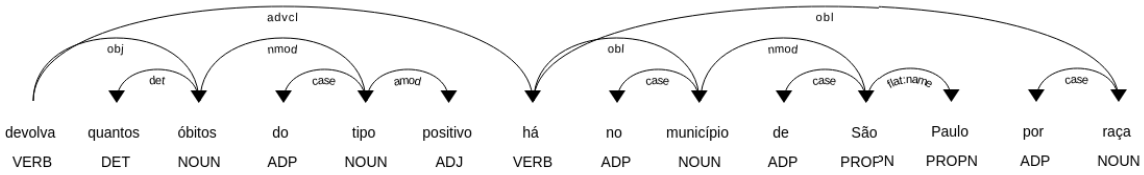


Figura 5.17: *Diagrama de dependências da consulta "Devolva quantos óbitos do tipo positivo há no município de São Paulo por raça", mencionada na seção 5.4.6*

Natural Language Query: devolva quantos óbitos do tipo positivo há no município de São Paulo por raça**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'óbitos', 'NOUN', 1, 'devolva', 'obj']
[3, 'do', 'ADP', 4, 'tipo', 'case']
[4, 'tipo', 'NOUN', 2, 'óbitos', 'nmod']
[5, 'positivo', 'ADJ', 4, 'tipo', 'amod']
[6, 'há', 'VERB', 1, 'devolva', 'advcl']
[7, 'no', 'ADP', 8, 'município', 'case']
[8, 'município', 'NOUN', 6, 'há', 'obl']
[9, 'de', 'ADP', 10, 'São', 'case']
[10, 'São', 'PROPN', 8, 'município', 'nmod']
[11, 'Paulo', 'PROPN', 10, 'São', 'flat:name']
[12, 'raça', 'VERB', 1, 'devolva', 'conj']

Parse Tree:

(134)ROOT
  (135)devolva
    (136)óbitos
      (137)tipo
        (138)do
        (139)positivo
    (140)há
      (141)município
        (142)no
        (143)São
          (144)de
          (145)Paulo
  (146)raça

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_obito']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_raca_cor']
pre_prepared_query = 'devolva óbitos do tipo positivo há no município de São Paulo raça'

NLIDB Interface Properties

group_ranking_to_df:
óbitos:casos.obito:NT;positivo:casos.obito:VT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.obito as casos_obito, casos.raca_cor as casos_raca_cor FROM casos WHERE casos.obito LIKE "%positivo%" AND casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_obito']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT casos_raca_cor, count(casos_obito) as count_casos_obito FROM NLIDB_result_set GROUP BY casos_raca_cor ORDER BY casos_raca_cor LIMIT 100'

Results

	casos_raca_cor	count_casos_obito
0	BRANCA	22
1	IGNORADO	10
2	NONE	2
3	PARDA	7
4	PRETA	2

Figura 5.18: *Resultado simplificado do processamento da consulta "Devolva quantos óbitos do tipo positivo há no município de São Paulo por raça", mencionada na seção 5.4.6*

Por diagnóstico de COVID-19

- Usando COUNT:
  - **Devolva quantos óbitos do tipo positivo há no município de São Paulo por diagnóstico**: ver figuras A.37 e A.38.
  - **Devolva quantos diagnóstico há agrupados por raça e por sexo e por diagnóstico**: ver figuras 5.19 e 5.20.
- Usando MAX:

  Devolva a máxima idade no município de São Paulo por diagnóstico: ver figuras A.39 e A.40.
- Usando MIN:

  Devolva a mínima idade no município de São Paulo por diagnóstico: ver figuras A.41 e A.42.
- Usando SUM:

  Devolva as idades somadas no município de São Paulo por diagnóstico: ver figuras A.43 e A.44.
- Usando AVG:

  Devolva a idade média no município de São Paulo por diagnóstico: ver figuras A.45 e A.46.

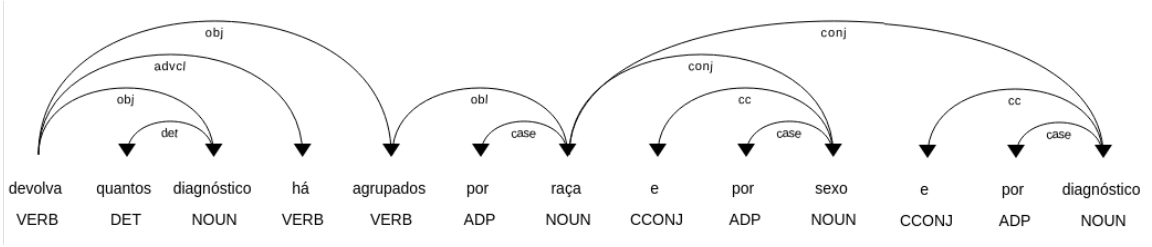


Figura 5.19: *Diagrama de dependências da consulta "Devolva quantos diagnóstico há agrupados por raça e por sexo e por diagnóstico", mencionada na seção 5.4.6*

Natural Language Query: devolva quantos diagnóstico há agrupados por raça e por sexo e por diagnóstico

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'diagnóstico', 'NOUN', 1, 'devolva', 'obj']
[3, 'há', 'VERB', 1, 'devolva', 'advcl']
[4, 'agrupados', 'g', 0, 'ROOT', 'root']
[5, 'raça', 'NOUN', 4, 'agrupados', 'obj']
[6, 'e', 'CCONJ', 7, 'sexo', 'cc']
[7, 'sexo', 'NOUN', 5, 'raça', 'conj']
[8, 'e', 'CCONJ', 2, 'diagnóstico', 'cc']
[9, 'diagnóstico', 'ADJ', 5, 'raça', 'conj']

Parse Tree:

(287)ROOT
  (288)devolva
    (290)diagnóstico
      (295)e
    (291)há
  (289)agrupados
    (292)raça
    (293)sexo
      (294)e
    (296)diagnóstico

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_diagnostico_covid19']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos', 'por', 'por', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_raca_cor', 'casos_cs_sexo', 'casos_diagnostico_covid19']
pre_prepared_query = 'devolva diagnóstico há agrupados raça e sexo e diagnóstico'

NLIDB Interface Properties

group_ranking_to_df: diagnóstico:casos.diagnostico_covid19:NT;raça:casos.nome_munic:VT;sexo:casos.cs_sexo:NT;diagnóstico:casos.diagnostico_covid19:NT;

nlidb_interface_sql = 'SELECT casos.diagnostico_covid19 as casos_diagnostico_covid19, casos.raca_cor as casos_raca_cor, casos.cs_sexo as casos_cs_sexo FROM casos WHERE casos.nome_munic LIKE "%raça%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_diagnostico_covid19']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT casos_raca_cor, casos_cs_sexo, casos_diagnostico_covid19, count(casos_diagnostico_covid19) as count_casos_diagnostico_covid19 FROM NLIDB_result_set GROUP BY casos_raca_cor, casos_cs_sexo, casos_diagnostico_covid19 ORDER BY casos_raca_cor, casos_cs_sexo, casos_diagnostico_covid19 LIMIT 100'

Results

	casos_raca_cor	casos_cs_sexo	casos_diagnostico_covid19	count_casos_diagnostico_covid19
0	BRANCA	FEMININO	CONFIRMADO	8
1	BRANCA	FEMININO	DESCARTADO	5
2	BRANCA	MASCULINO	CONFIRMADO	8
3	BRANCA	MASCULINO	DESCARTADO	8
4	IGNORADO	FEMININO	CONFIRMADO	3
5	IGNORADO	FEMININO	DESCARTADO	3
6	IGNORADO	MASCULINO	DESCARTADO	2
7	PARDA	FEMININO	CONFIRMADO	3
8	PARDA	FEMININO	DESCARTADO	2
9	PARDA	MASCULINO	CONFIRMADO	1
10	PARDA	MASCULINO	DESCARTADO	4
11	PRETA	FEMININO	DESCARTADO	2
12	PRETA	MASCULINO	CONFIRMADO	1

Figura 5.20: *Resultado simplificado do processamento da consulta "Devolva quantos diagnóstico há agrupados por raça e por sexo e por diagnóstico", mencionada na seção 5.4.6*

5.5 Discussão

É possível notar que, de maneira geral, o conteúdo de “nlidb_interface_sql” e de “pos_glamorise_sql”, ou seja, os comandos SQL resultantes do mapeamento, são condizentes com as consultas testadas.

A sutil diferença na escrita das consultas nas figuras 5.4 e 5.6 não impactou no resultado final, visto que “nlidb_interface_sql” e “pos_glamorise_sql” permaneceram os mesmos. Um caso parecido ocorre com as consultas nas imagens 5.12 e 5.14.

Entretanto, para as consultas nas figuras 5.8 e 5.10, nota-se diferença no resultado do processamento. Na figura 5.8, o sistema não diferencia os óbitos realmente ocorridos (casos.obito = “POSITIVO”) dos óbitos não ocorridos (casos.obito = “NEGATIVO”), retornando, assim, a quantidade total de registros da tabela “casos”. Na imagem 5.10, em vez de retornar apenas a quantidade total de casos, o sistema devolveu esse número agrupado pelos três valores possíveis do atributo “cs_sexo” em “casos”, pois a palavra “casos” foi mapeada para esse atributo, conforme mostrado no conteúdo de “group_ranking_to_df”. No entanto, somando o valor númerico de cada grupo, o resultado é a quantidade total de registros em “casos”, como mostrada na figura 5.8. Algo semelhante ocorre na imagem 5.16. Exemplos de uso do valor “POSITIVO” no atributo “casos.obito” estão nas figuras 5.12 e 5.18.

Um caso peculiar é exibido na imagem 5.20, em que há agrupamento por três variáveis e cujo resultado satisfatório era inesperado, devido à complexidade da consulta. Além disso, na consulta, o termo “diagnóstico” no singular foi proposital, pois, com esse termo no plural, o resultado não foi satisfatório.

Capítulo 6

Conclusão

Primeiramente, foi necessário compreender o projeto original descrito em Novello, 2021, para determinar as adaptações que deveriam ser feitas, com o objetivo de realizar experimentos com frases em Português.

Por exemplo, foi preciso pesquisar alternativas para o NLTK, visto que, durante o desenvolvimento desta monografia, não foram encontradas determinadas funções para textos escritos em Português. Uma dessas funções importantes é a que devolve o lema de uma palavra.

Durante a pesquisa, encontrou-se a biblioteca *Spacy*, que identifica, em frases escritas em Português, a classe gramatical e a relação de dependência das palavras, além de devolver o lema de cada termo da frase. É uma biblioteca bastante interessante e com diversas outras funções para a língua portuguesa.

Um algoritmo precisou ser implementado, com o intuito de substituir a biblioteca *Stanford Parser*, que não possui versão para a língua portuguesa.

Após escolher a base de dados da Fundação SEADE sobre casos de COVID-19, pensou-se em 32 frases, algumas com agrupamento, outras com agregação, outras com os dois e outras com nenhum dos dois, a fim de testar o sistema adaptado NaLIR + GLAMORISE.

Os resultados obtidos foram satisfatórios, com alguns casos peculiares, conforme relatado na seção 5.5. A fim de obter bons resultados, recomenda-se que a grafia das palavras da consulta esteja correta, assim como a pontuação e os acentos, pois, por exemplo, dependendo da grafia, da pontuação e dos acentos, a consulta pode ficar ambígua ou torna-se maior o risco de uma interpretação errônea ou de um processamento incorreto da frase pelo sistema adaptado.

6.1 Disciplinas relacionadas ao trabalho

As seguintes disciplinas foram importantes para o desenvolvimento desta monografia:

- MAC0110 - Introdução à Computação

- MAC0121 - Algoritmos e Estruturas de Dados I
- MAC0216 - Técnicas de Programação I
- MAC0239 - Introdução à Lógica e Verificação de Programas
- MAC0323 - Algoritmos e Estruturas de Dados II
- MAC0338 - Análise de Algoritmos
- MAC0350 - Introdução ao Desenvolvimento de Sistemas de Software
- MAC0426 - Sistemas de Bancos de Dados
- MAC0439 - Laboratório de Banco de Dados

6.2 Trabalhos futuros

Este trabalho pode ser estendido, com a realização dos seguintes itens:

- Experimentos com banco de dados contendo mais de uma tabela, para possibiltar o mapeamento e a execução de consultas envolvendo junções;
- Redução do tempo de execução do processamento da consulta

Apêndice A

Figuras: relação de dependência e resultado do processamento no NaLIR / GLAMORISE

Neste apêndice, as imagens cuja numeração possui um número ímpar após o "A." são diagramas gerados pelo *Spacy* (Explosion, 2022b), que mostram, para cada consulta do capítulo 5, a relação de dependência, indicada nas arestas, e a classe gramatical das palavras, indicada na parte inferior do diagrama. Para saber o significado dos termos que indicam a relação de dependência e a classe gramatical, consultar Contributors, 2022a e Contributors, 2022b, respectivamente.

Abaixo de cada diagrama, há uma versão simplificada do resultado do processamento no NaLIR e no GLAMORISE, para a consulta mostrada no diagrama.

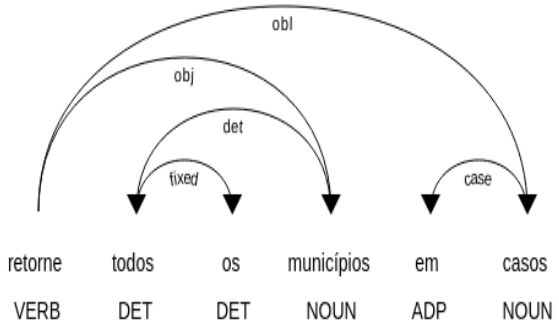
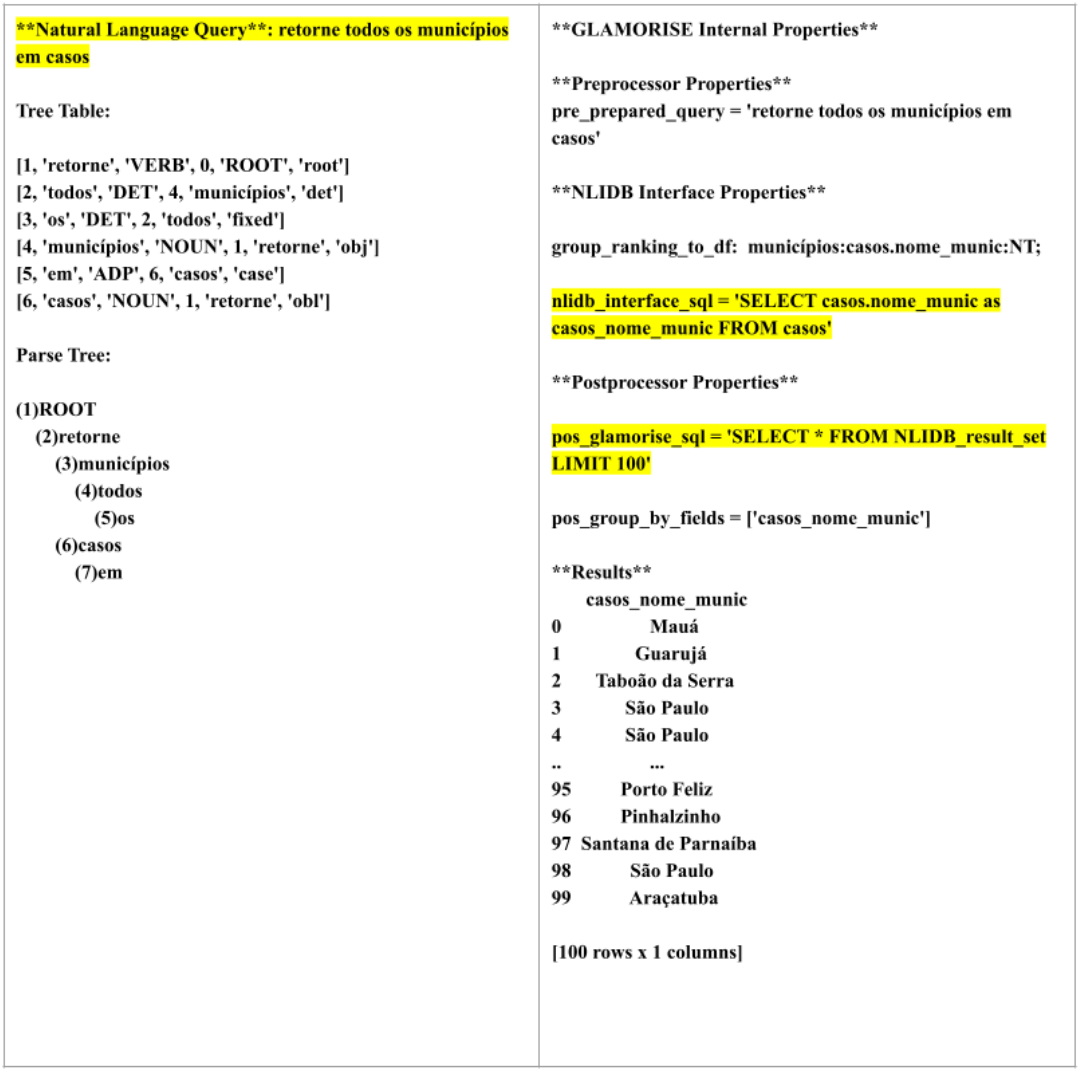


Figura A.1: *Diagrama de dependências da consulta "Retorne todos os municípios em casos", mencionada na seção 5.4.1*



Natural Language Query: retorne todos os municípios em casos

Tree Table:

[1, 'retorne', 'VERB', 0, 'ROOT', 'root']
[2, 'todos', 'DET', 4, 'municípios', 'det']
[3, 'os', 'DET', 2, 'todos', 'fixed']
[4, 'municípios', 'NOUN', 1, 'retorne', 'obj']
[5, 'em', 'ADP', 6, 'casos', 'case']
[6, 'casos', 'NOUN', 1, 'retorne', 'obl']

Parse Tree:

(1)ROOT
  (2)retorne
    (3)municípios
      (4)todos
        (5)os
    (6)casos
      (7)em

GLAMORISE Internal Properties

Preprocessor Properties
pre_prepared_query = 'retorne todos os municípios em casos'

NLIDB Interface Properties

group_ranking_to_df: municípios:casos.nome_munic:NT;

nlidb_interface_sql = 'SELECT casos.nome_munic as casos_nome_munic FROM casos'

Postprocessor Properties

pos_glamorise_sql = 'SELECT * FROM NLIDB_result_set LIMIT 100'

pos_group_by_fields = ['casos_nome_munic']

Results

	casos_nome_munic
0	Mauá
1	Guarujá
2	Taboão da Serra
3	São Paulo
4	São Paulo
..	...
95	Porto Feliz
96	Pinhalzinho
97	Santana de Parnaíba
98	São Paulo
99	Araçatuba

[100 rows x 1 columns]

Figura A.2: *Resultado simplificado do processamento da consulta "Retorne todos os municípios em casos", mencionada na seção 5.4.1*

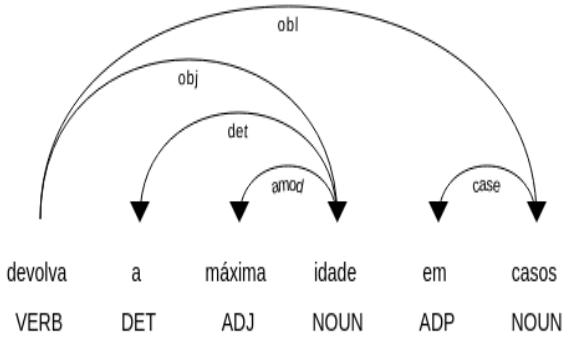


Figura A.3: *Diagrama de dependências da consulta "Devolva a máxima idade em casos", mencionada na seção 5.4.3*

****Natural Language Query**: devolva a máxima idade em casos**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'em', 'ADP', 5, 'casos', 'case']
[5, 'casos', 'NOUN', 1, 'devolva', 'obl']

Parse Tree:

(11)ROOT
(12)devolva
(13)idade
(14)a
(15)casos
(16)em

****GLAMORISE Internal Properties****

****Preprocessor Properties****
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['max']
pre_cut_text = ['máxima']
pre_prepared_query = 'devolva a idade em casos'

****NLIDB Interface Properties****

group_ranking_to_df: idade:casos.idade:NT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos'

****Postprocessor Properties****
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['max']

pos_glamorise_sql = 'SELECT max(casos_idade) as max_casos_idade FROM NLIDB_result_set LIMIT 100'

****Results****

	max_casos_idade
0	**121.0**

Figura A.4: *Resultado simplificado do processamento da consulta "Devolva a máxima idade em casos", mencionada na seção 5.4.3*

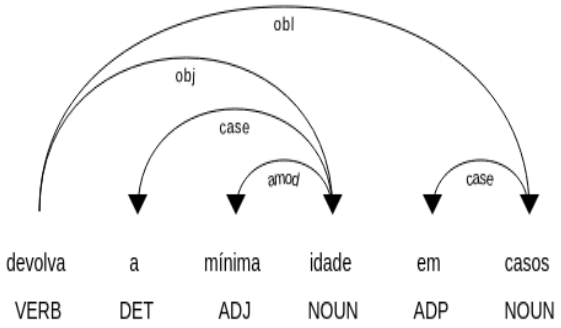


Figura A.5: *Diagrama de dependências da consulta "Devolva a mínima idade em casos", mencionada na seção 5.4.3*

****Natural Language Query**: devolva a mínima idade em casos**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'em', 'ADP', 5, 'casos', 'case']
[5, 'casos', 'NOUN', 1, 'devolva', 'obl']

Parse Tree:

(18)ROOT
(19)devolva
(20)idade
(21)a
(22)casos
(23)em

****GLAMORISE Internal Properties****

****Preprocessor Properties****
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['min']
pre_cut_text = ['mínima']
pre_prepared_query = 'devolva a idade em casos'

****NLIDB Interface Properties****

group_ranking_to_df: idade:casos.idade:NT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos'

****Postprocessor Properties****
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['min']

pos_glamorise_sql = 'SELECT min(casos_idade) as min_casos_idade FROM NLIDB_result_set LIMIT 100'

****Results****
min_casos_idade
0 0.0

Figura A.6: *Resultado simplificado do processamento da consulta "Devolva a mínima idade em casos", mencionada na seção 5.4.3*

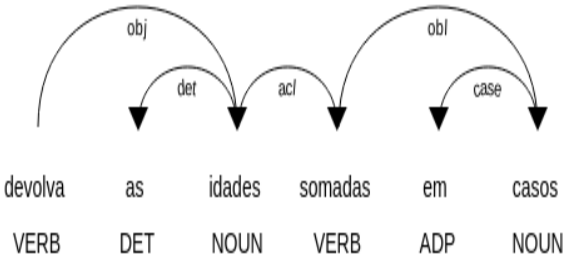


Figura A.7: *Diagrama de dependências da consulta "Devolva as idades somadas em casos", mencionada na seção 5.4.3*

Natural Language Query: devolva as idades somadas em casos

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'as', 'DET', 3, 'idades', 'det']
[3, 'idades', 'NOUN', 1, 'devolva', 'obj']
[4, 'em', 'ADP', 5, 'casos', 'case']
[5, 'casos', 'NOUN', 1, 'devolva', 'obl']

Parse Tree:

(25)ROOT
  (26)devolva
    (27)idades
      (28)as
    (29)casos
      (30)em

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['sum']
pre_cut_text = ['somadas']
pre_prepared_query = 'devolva as idades em casos'

NLIDB Interface Properties

group_ranking_to_df: idades:casos.idade:NT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['sum']

pos_glamorise_sql = 'SELECT sum(casos_idade) as sum_casos_idade FROM NLIDB_result_set LIMIT 100'

Results
  sum_casos_idade
0 196639.0

Figura A.8: *Resultado simplificado do processamento da consulta "Devolva as idades somadas em casos", mencionada na seção 5.4.3*

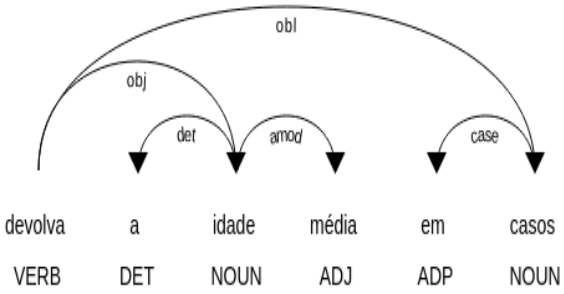


Figura A.9: *Diagrama de dependências da consulta "Devolva a idade média em casos", mencionada na seção 5.4.3*

Natural Language Query: devolva a idade média em casos**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'em', 'ADP', 5, 'casos', 'case']
[5, 'casos', 'NOUN', 1, 'devolva', 'obl']

Parse Tree:

(32)ROOT
  (33)devolva
    (34)idade
      (35)a
    (36)casos
      (37)em

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['avg']
pre_cut_text = ['média']
pre_prepared_query = 'devolva a idade em casos'

NLIDB Interface Properties

group_ranking_to_df: idade:casos.idade:NT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['avg']

pos_glamorise_sql = 'SELECT avg(casos_idade) as avg_casos_idade FROM NLIDB_result_set LIMIT 100'

Results

	avg_casos_idade
0	40.228928

Figura A.10: *Resultado simplificado do processamento da consulta "Devolva a idade média em casos", mencionada na seção 5.4.3*

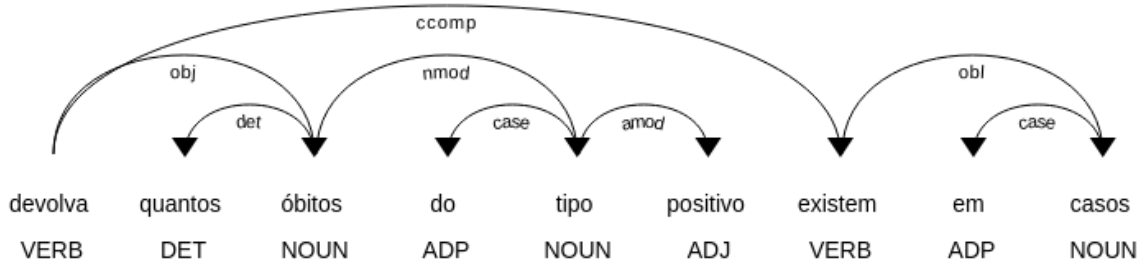


Figura A.11: *Diagrama de dependências da consulta "Devolva quantos óbitos do tipo positivo existem em casos", mencionada na seção 5.4.4*

Natural Language Query: devolva quantos óbitos do tipo positivo existem em casos**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'óbitos', 'NOUN', 1, 'devolva', 'obj']
[3, 'do', 'ADP', 4, 'tipo', 'case']
[4, 'tipo', 'NOUN', 2, 'óbitos', 'nmod']
[5, 'positivo', 'ADJ', 4, 'tipo', 'amod']
[6, 'existem', 'VERB', 1, 'devolva', 'advcl']
[7, 'em', 'ADP', 8, 'casos', 'case']
[8, 'casos', 'NOUN', 6, 'existem', 'obl']

Parse Tree:

(1)ROOT
  (2)devolva
    (3)óbitos
      (4)tipo
        (5)do
        (6)positivo
    (7)existem
      (8)casos
        (9)em

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_obito']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos']
pre_prepared_query = 'devolva óbitos do tipo positivo existem em casos'

NLIDB Interface Properties

group_ranking_to_df:
óbitos:casos.obito:NT;positivo:casos.obito:VT;

nlidb_interface_sql = 'SELECT casos.obito as casos_obito FROM casos WHERE casos.obito LIKE "%positivo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_obito']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT count(casos_obito) as count_casos_obito FROM NLIDB_result_set LIMIT 100'

Results
  count_casos_obito
0 136

Figura A.12: *Resultado simplificado do processamento da consulta "Devolva quantos óbitos do tipo positivo existem em casos", mencionada na seção 5.4.4*

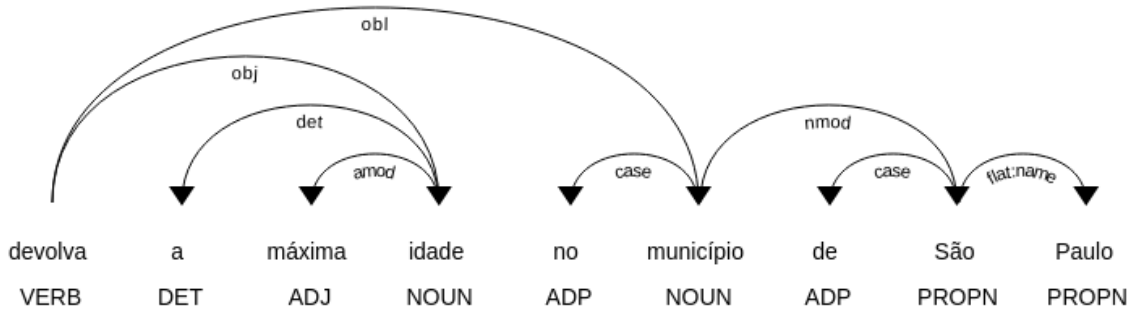


Figura A.13: *Diagrama de dependências da consulta "Devolva a máxima idade no município de São Paulo", mencionada na seção 5.4.4*

Natural Language Query: devolva a máxima idade no município de São Paulo**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']

Parse Tree:

(55)ROOT
  (56)devolva
    (57)idade
      (58)a
    (59)município
      (60)no
      (61)São
        (62)de
        (63)Paulo

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['max']
pre_cut_text = ['máxima']
pre_prepared_query = 'devolva a idade no município de São Paulo'

NLIDB Interface Properties

group_ranking_to_df:
idade:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['max']

pos_glamorise_sql = 'SELECT max(casos_idade) as max_casos_idade FROM NLIDB_result_set LIMIT 100'

Results
  max_casos_idade
0 121.0

Figura A.14: *Resultado simplificado do processamento da consulta "Devolva a máxima idade no município de São Paulo", mencionada na seção 5.4.4*

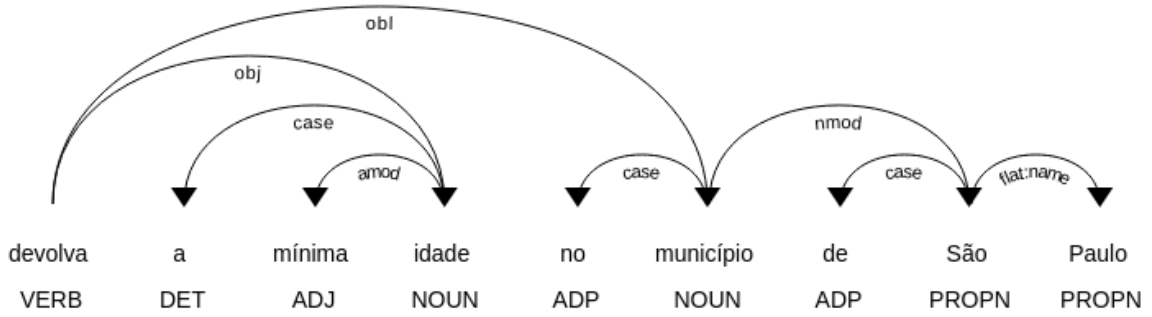


Figura A.15: *Diagrama de dependências da consulta "Devolva a mínima idade no município de São Paulo", mencionada na seção 5.4.4*

Natural Language Query: devolva a mínima idade no município de São Paulo**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']

Parse Tree:

(65)ROOT
  (66)devolva
    (67)idade
      (68)a
    (69)município
      (70)no
      (71)São
        (72)de
        (73)Paulo

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['min']
pre_cut_text = ['mínima']
pre_prepared_query = 'devolva a idade no município de São Paulo'

NLIDB Interface Properties

group_ranking_to_df:
idade:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['min']

pos_glamorise_sql = 'SELECT min(casos_idade) as min_casos_idade FROM NLIDB_result_set LIMIT 100'

Results
  min_casos_idade
0 0.0

Figura A.16: *Resultado simplificado do processamento da consulta "Devolva a mínima idade no município de São Paulo", mencionada na seção 5.4.4*

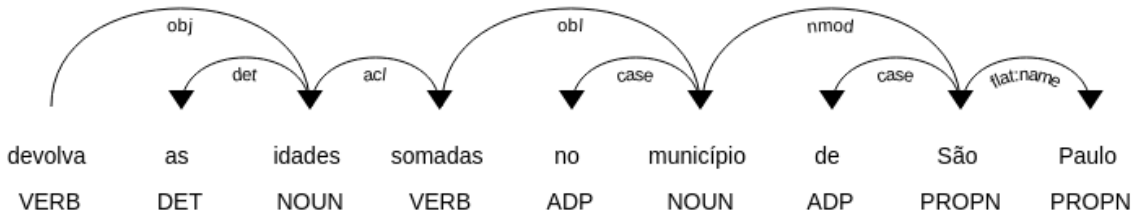


Figura A.17: *Diagrama de dependências da consulta "Devolva as idades somadas no município de São Paulo", mencionada na seção 5.4.4*

Natural Language Query: devolva as idades somadas no município de São Paulo**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'as', 'DET', 3, 'idades', 'det']
[3, 'idades', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']

Parse Tree:

(75)ROOT
  (76)devolva
    (77)idades
      (78)as
    (79)município
      (80)no
      (81)São
        (82)de
        (83)Paulo

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['sum']
pre_cut_text = ['somadas']
pre_prepared_query = 'devolva as idades no município de São Paulo'

NLIDB Interface Properties

group_ranking_to_df:
idades:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['sum']

pos_glamorise_sql = 'SELECT sum(casos_idade) as sum_casos_idade FROM NLIDB_result_set LIMIT 100'

Results
  sum_casos_idade
0 38793.0

Figura A.18: *Resultado simplificado do processamento da consulta "Devolva as idades somadas no município de São Paulo", mencionada na seção 5.4.4*

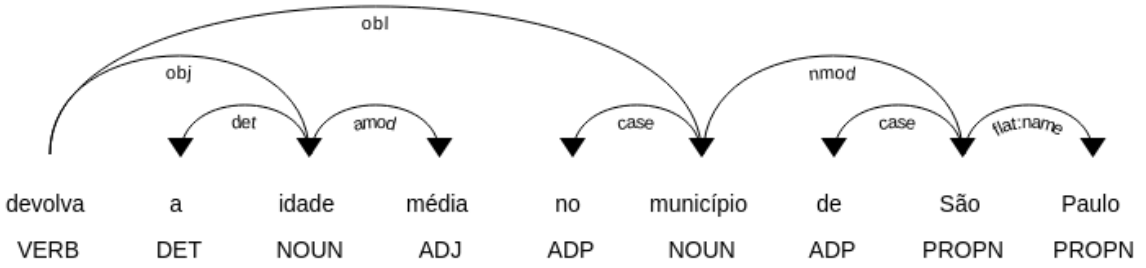


Figura A.19: *Diagrama de dependências da consulta "Devolva a idade média no município de São Paulo", mencionada na seção 5.4.4*

Natural Language Query: devolva a idade média no município de São Paulo**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']

Parse Tree:

(85)ROOT
  (86)devolva
    (87)idade
      (88)a
    (89)município
      (90)no
      (91)São
        (92)de
        (93)Paulo

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['avg']
pre_cut_text = ['média']
pre_prepared_query = 'devolva a idade no município de São Paulo'

NLIDB Interface Properties

group_ranking_to_df:
idade:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['avg']

pos_glamorise_sql = 'SELECT avg(casos_idade) as avg_casos_idade FROM NLIDB_result_set LIMIT 100'

Results
  avg_casos_idade
0   41.0074

Figura A.20: *Resultado simplificado do processamento da consulta "Devolva a idade média no município de São Paulo", mencionada na seção 5.4.4*

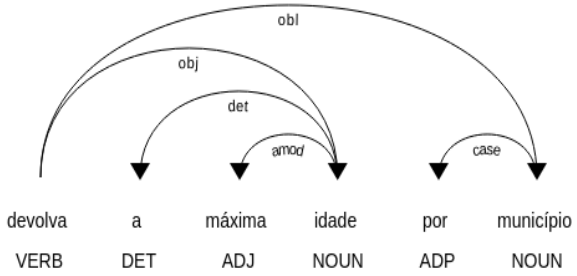


Figura A.21: *Diagrama de dependências da consulta "Devolva a máxima idade por município", mencionada na seção 5.4.5*

Natural Language Query: devolva a máxima idade por município

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'município', 'NOUN', 1, 'devolva', 'obj']

Parse Tree:

(110)ROOT
  (111)devolva
    (112)idade
      (113)a
    (114)município

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['max']
pre_cut_text = ['máxima', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_nome_munic']
pre_prepared_query = 'devolva a idade município'

NLIDB Interface Properties
group_ranking_to_df:
idade:casos.idade:NT;município:casos.nome_munic:NT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade, casos.nome_munic as casos_nome_munic FROM casos'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['max']

pos_glamorise_sql = 'SELECT casos_nome_munic, max(casos_idade) as max_casos_idade FROM NLIDB_result_set GROUP BY casos_nome_munic ORDER BY casos_nome_munic LIMIT 100'

Results

	casos_nome_munic	max_casos_idade
0	Adamantina	49.0
1	Aguaí	50.0
2	Agudos	59.0
3	Alambari	31.0
4	Altair	8.0
..	...	...
95	Coronel Macedo	38.0
96	Cosmorama	36.0
97	Cosmópolis	71.0
98	Cotia	81.0
99	Cravinhos	68.0

[100 rows x 2 columns]

Figura A.22: *Resultado simplificado do processamento da consulta "Devolva a máxima idade por município", mencionada na seção 5.4.5*

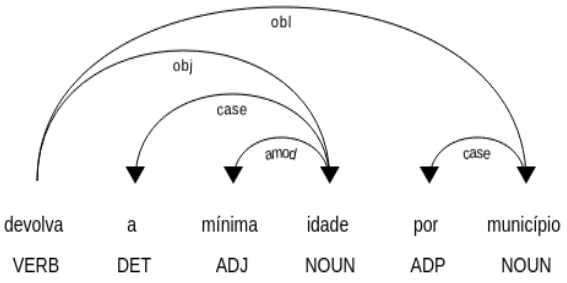


Figura A.23: *Diagrama de dependências da consulta "Devolva a mínima idade por município", mencionada na seção 5.4.5*

****Natural Language Query**: devolva a mínima idade por município**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'município', 'NOUN', 1, 'devolva', 'obj']

Parse Tree:

(116)ROOT
(117)devolva
(118)idade
(119)a
(120)município

****GLAMORISE Internal Properties****

****Preprocessor Properties****
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['min']
pre_cut_text = ['mínima', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_nome_munic']
pre_prepared_query = 'devolva a idade município'

****NLIDB Interface Properties****

group_ranking_to_df:
idade:casos.idade:NT;município:casos.nome_munic:NT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade, casos.nome_munic as casos_nome_munic FROM casos'

****Postprocessor Properties****
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['min']

pos_glamorise_sql = 'SELECT casos_nome_munic, min(casos_idade) as min_casos_idade FROM NLIDB_result_set GROUP BY casos_nome_munic ORDER BY casos_nome_munic LIMIT 100'

****Results****

	casos_nome_munic	min_casos_idade
0	Adamantina	20.0
1	Aguaí	23.0
2	Agudos	10.0
3	Alambari	31.0
4	Altair	8.0
..	...	...
95	Coronel Macedo	38.0
96	Cosmorama	35.0
97	Cosmópolis	17.0
98	Cotia	17.0
99	Cravinhos	34.0

[100 rows x 2 columns]

Figura A.24: *Resultado simplificado do processamento da consulta "Devolva a mínima idade por município", mencionada na seção 5.4.5*

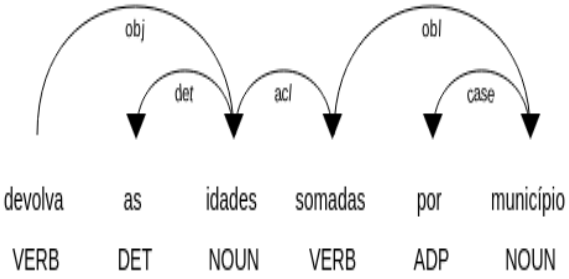


Figura A.25: *Diagrama de dependências da consulta "Devolva as idades somadas por município", mencionada na seção 5.4.5*

****Natural Language Query**: devolva as idades somadas por município**

Tree Table:

```
[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'as', 'DET', 3, 'idades', 'det']
[3, 'idades', 'NOUN', 1, 'devolva', 'obj']
[4, 'município', 'NOUN', 1, 'devolva', 'obj']
```

Parse Tree:

```
(122)ROOT
   (123)devolva
      (124)idades
         (125)as
      (126)município
```

****GLAMORISE Internal Properties****

****Preprocessor Properties****

```
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['sum']
pre_cut_text = ['somadas', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_nome_munic']
pre_prepared_query = 'devolva as idades município'
```

****NLIDB Interface Properties****

```
group_ranking_to_df:
idades:casos.idade:NT;município:casos.nome_munic:NT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade, casos.nome_munic
as casos_nome_munic FROM casos'
```

****Postprocessor Properties****

```
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['sum']

pos_glamorise_sql = 'SELECT casos_nome_munic, sum(casos_idade) as
sum_casos_idade FROM NLIDB_result_set GROUP BY casos_nome_munic
ORDER BY casos_nome_munic LIMIT 100'
```

****Results****

	casos_nome_munic	sum_casos_idade
0	Adamantina	216.0
1	Aguaí	257.0
2	Agudos	122.0
3	Alambari	31.0
4	Altair	8.0
..	. ..	...
95	Coronel Macedo	38.0
96	Cosmorama	71.0
97	Cosmópolis	402.0
98	Cotia	979.0
99	Cravinhos	102.0

[100 rows x 2 columns]

Figura A.26: *Resultado simplificado do processamento da consulta "Devolva as idades somadas por município", mencionada na seção 5.4.5*

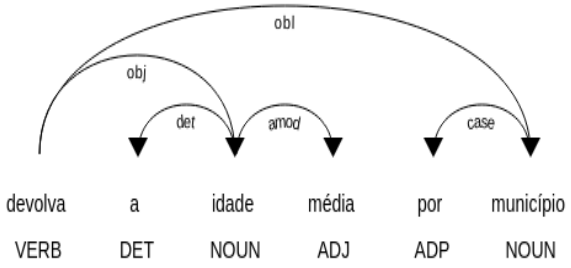


Figura A.27: *Diagrama de dependências da consulta "Devolva a idade média por município", mencionada na seção 5.4.5*

****Natural Language Query**: devolva a idade média por município**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'município', 'NOUN', 1, 'devolva', 'obj']

Parse Tree:

(128)ROOT
(129)devolva
(130)idade
(131)a
(132)município

****GLAMORISE Internal Properties****

****Preprocessor Properties****
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['avg']
pre_cut_text = ['média', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_nome_munic']
pre_prepared_query = 'devolva a idade município'

****NLIDB Interface Properties****

group_ranking_to_df:
idade:casos.idade:NT;município:casos.nome_munic:NT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade, casos.nome_munic as casos_nome_munic FROM casos'

****Postprocessor Properties****
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['avg']

pos_glamorise_sql = 'SELECT casos_nome_munic, avg(casos_idade) as avg_casos_idade FROM NLIDB_result_set GROUP BY casos_nome_munic ORDER BY casos_nome_munic LIMIT 100'

****Results****

	casos_nome_munic	avg_casos_idade
0	Adamantina	36.000000
1	Aguaí	36.714286
2	Agudos	30.500000
3	Alambari	31.000000
4	Altair	8.000000
..	...	...
95	Coronel Macedo	38.000000
96	Cosmorama	35.500000
97	Cosmópolis	40.200000
98	Cotia	48.950000
99	Cravinhos	51.000000

[100 rows x 2 columns]

Figura A.28: *Resultado simplificado do processamento da consulta "Devolva a idade média por município", mencionada na seção 5.4.5*

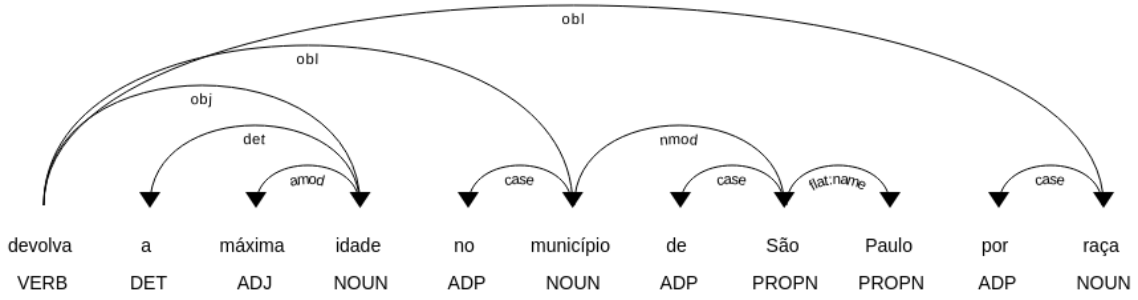


Figura A.29: *Diagrama de dependências da consulta "Devolva a máxima idade no município de São Paulo por raça", mencionada na seção 5.4.6*

Natural Language Query: devolva a máxima idade no município de São Paulo por raça**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']
[9, 'raça', 'VERB', 1, 'devolva', 'ccomp']

Parse Tree:

(148)ROOT
  (149)devolva
    (150)idade
      (151)a
    (152)município
      (153)no
      (154)São
        (155)de
        (156)Paulo
    (157)raça

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['max']
pre_cut_text = ['máxima', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_raca_cor']
pre_prepared_query = 'devolva a idade no município de São Paulo raça'

NLIDB Interface Properties

group_ranking_to_df: idade:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade, casos.raca_cor as casos_raca_cor FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['max']

pos_glamorise_sql = 'SELECT casos_raca_cor, max(casos_idade) as max_casos_idade FROM NLIDB_result_set GROUP BY casos_raca_cor ORDER BY casos_raca_cor LIMIT 100'

Results

	casos_raca_cor	max_casos_idade
0	None	120.0
1	AMARELA	90.0
2	BRANCA	95.0
3	IGNORADO	121.0
4	INDIGENA	67.0
5	NONE	73.0
6	PARDA	90.0
7	PRETA	77.0

Figura A.30: *Resultado simplificado do processamento da consulta "Devolva a máxima idade no município de São Paulo por raça", mencionada na seção 5.4.6*

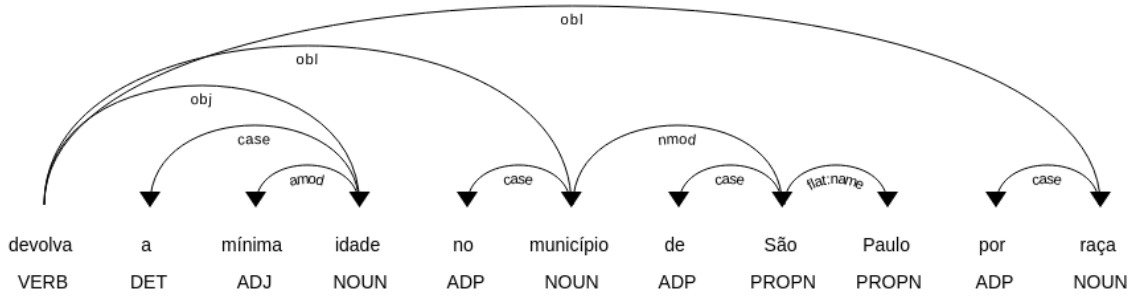


Figura A.31: *Diagrama de dependências da consulta "Devolva a mínima idade no município de São Paulo por raça", mencionada na seção 5.4.6*

****Natural Language Query**: devolva a mínima idade no município de São Paulo por raça**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']
[9, 'raça', 'VERB', 1, 'devolva', 'ccomp']

Parse Tree:

(159)ROOT
  (160)devolva
    (161)idade
      (162)a
    (163)município
      (164)no
      (165)São
        (166)de
        (167)Paulo
    (168)raça

****GLAMORISE Internal Properties****

****Preprocessor Properties****
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['min']
pre_cut_text = ['mínima', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_raca_cor']
pre_prepared_query = 'devolva a idade no município de São Paulo raça'

****NLIDB Interface Properties****

group_ranking_to_df: idade:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade, casos.raca_cor as casos_raca_cor FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

****Postprocessor Properties****
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['min']

pos_glamorise_sql = 'SELECT casos_raca_cor, min(casos_idade) as min_casos_idade FROM NLIDB_result_set GROUP BY casos_raca_cor ORDER BY casos_raca_cor LIMIT 100'

****Results****

	casos_raca_cor	min_casos_idade
0	None	12.0
1	AMARELA	20.0
2	BRANCA	0.0
3	IGNORADO	0.0
4	INDIGENA	24.0
5	NONE	3.0
6	PARDA	0.0
7	PRETA	2.0

Figura A.32: *Resultado simplificado do processamento da consulta "Devolva a mínima idade no município de São Paulo por raça", mencionada na seção 5.4.6*

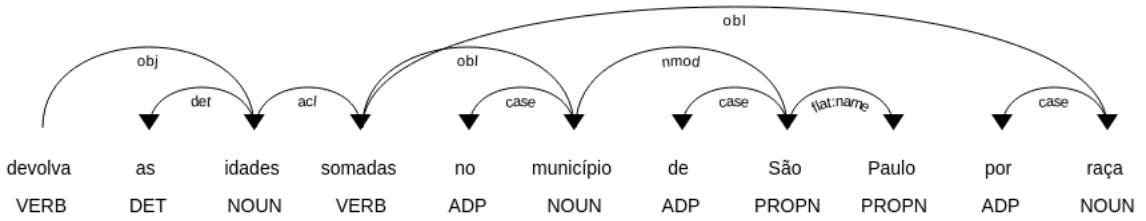


Figura A.33: *Diagrama de dependências da consulta "Devolva as idades somadas no município de São Paulo por raça", mencionada na seção 5.4.6*

Natural Language Query: devolva as idades somadas no município de São Paulo por raça**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'as', 'DET', 3, 'idades', 'det']
[3, 'idades', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']
[9, 'raça', 'VERB', 1, 'devolva', 'ccomp']

Parse Tree:

(170)ROOT
  (171)devolva
    (172)idades
      (173)as
    (174)município
      (175)no
      (176)São
        (177)de
        (178)Paulo
    (179)raça

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['sum']
pre_cut_text = ['somadas', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_raca_cor']
pre_prepared_query = 'devolva as idades no município de São Paulo raça'

NLIDB Interface Properties

group_ranking_to_df: idades:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade, casos.raca_cor as casos_raca_cor FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['sum']

pos_glamorise_sql = 'SELECT casos_raca_cor, sum(casos_idade) as sum_casos_idade FROM NLIDB_result_set GROUP BY casos_raca_cor ORDER BY casos_raca_cor LIMIT 100'

Results

	casos_raca_cor	sum_casos_idade
0	None	592.0
1	AMARELA	1156.0
2	BRANCA	15428.0
3	IGNORADO	8065.0
4	INDIGENA	133.0
5	NONE	618.0
6	PARDA	11028.0
7	PRETA	1773.0

Figura A.34: *Resultado simplificado do processamento da consulta "Devolva as idades somadas no município de São Paulo por raça", mencionada na seção 5.4.6*

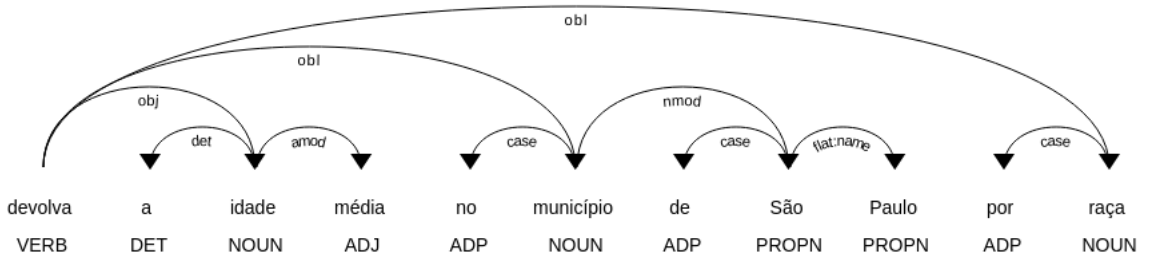


Figura A.35: *Diagrama de dependências da consulta "Devolva a idade média no município de São Paulo por raça", mencionada na seção 5.4.6*

Natural Language Query: devolva a idade média no município de São Paulo por raça**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']
[9, 'raça', 'VERB', 1, 'devolva', 'ccomp']

Parse Tree:

(181)ROOT
  (182)devolva
    (183)idade
      (184)a
    (185)município
      (186)no
      (187)São
        (188)de
        (189)Paulo
    (190)raça

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['avg']
pre_cut_text = ['média', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_raca_cor']
pre_prepared_query = 'devolva a idade no município de São Paulo raça'

NLIDB Interface Properties

group_ranking_to_df: idade:casos.idade:NT;paulo:casos.nome_munic:VT;

nlidb_interface_sql = 'SELECT casos.idade as casos_idade, casos.raca_cor as casos_raca_cor FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['avg']

pos_glamorise_sql = 'SELECT casos_raca_cor, avg(casos_idade) as avg_casos_idade FROM NLIDB_result_set GROUP BY casos_raca_cor ORDER BY casos_raca_cor LIMIT 100'

Results

	casos_raca_cor	avg_casos_idade
0	None	53.818182
1	AMARELA	41.285714
2	BRANCA	41.361930
3	IGNORADO	42.005208
4	INDIGENA	44.333333
5	NONE	41.200000
6	PARDA	39.669065
7	PRETA	38.543478

Figura A.36: *Resultado simplificado do processamento da consulta "Devolva a idade média no município de São Paulo por raça", mencionada na seção 5.4.6*

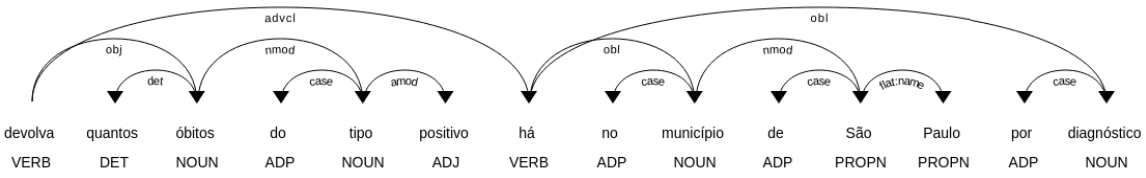


Figura A.37: *Diagrama de dependências da consulta "Devolva quantos óbitos do tipo positivo há no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

Natural Language Query: devolva quantos óbitos do tipo positivo há no município de São Paulo por diagnóstico**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'óbitos', 'NOUN', 1, 'devolva', 'obj']
[3, 'do', 'ADP', 4, 'tipo', 'case']
[4, 'tipo', 'NOUN', 2, 'óbitos', 'nmod']
[5, 'positivo', 'ADJ', 4, 'tipo', 'amod']
[6, 'há', 'VERB', 1, 'devolva', 'advcl']
[7, 'no', 'ADP', 8, 'município', 'case']
[8, 'município', 'NOUN', 6, 'há', 'obl']
[9, 'de', 'ADP', 10, 'São', 'case']
[10, 'São', 'PROPN', 8, 'município', 'nmod']
[11, 'Paulo', 'PROPN', 10, 'São', 'flat:name']
[12, 'diagnóstico', 'ADJ', 10, 'São', 'amod']

Parse Tree:

(192)ROOT
  (193)devolva
    (194)óbitos
      (195)tipo
        (196)do
        (197)positivo
    (198)há
      (199)município
        (200)no
        (201)São
          (202)de
          (203)Paulo
          (204)diagnóstico

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_obito']
pre_aggregation_functions = ['count']
pre_cut_text = ['quantos', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_diagnostico_covid19']
pre_prepared_query = 'devolva óbitos do tipo positivo há no município de São Paulo diagnóstico'

NLIDB Interface Properties

group_ranking_to_df:
óbitos:casos.obito:NT;positivo:casos.obito:VT;paulo:casos.nome_munic:VT;diagnóstico:casos.diagnostico_covid19:NT;

nlidb_interface_sql = 'SELECT casos.obito as casos_obito, casos.diagnostico_covid19 as casos_diagnostico_covid19 FROM casos WHERE casos.nome_munic LIKE "%Paulo%" AND casos.obito LIKE "%positivo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_obito']
pos_aggregation_functions = ['count']

pos_glamorise_sql = 'SELECT casos_diagnostico_covid19, count(casos_obito) as count_casos_obito FROM NLIDB_result_set GROUP BY casos_diagnostico_covid19 ORDER BY casos_diagnostico_covid19 LIMIT 100'

Results

	casos_diagnostico_covid19	count_casos_obito
0	CONFIRMADO	24
1	DESCARTADO	19

Figura A.38: *Resultado simplificado do processamento da consulta "Devolva quantos óbitos do tipo positivo há no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

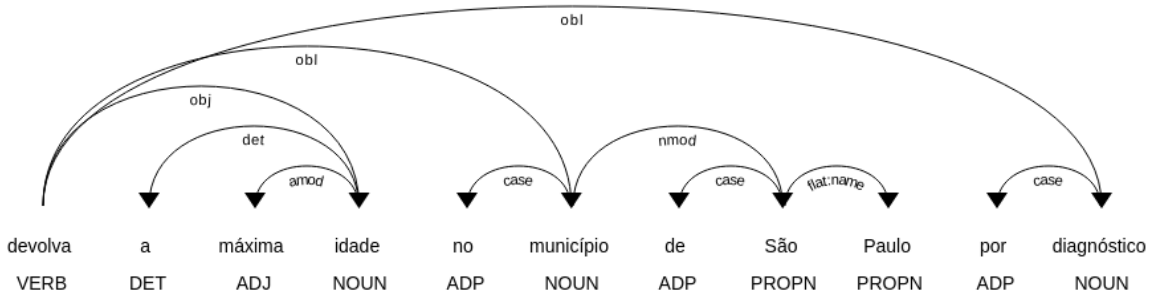


Figura A.39: *Diagrama de dependências da consulta "Devolva a máxima idade no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

Natural Language Query: devolva a máxima idade no município de São Paulo por diagnóstico

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']
[9, 'diagnóstico', 'ADJ', 7, 'São', 'amod']

Parse Tree:

(206)ROOT
  (207)devolva
    (208)idade
      (209)a
    (210)município
      (211)no
      (212)São
        (213)de
        (214)Paulo
        (215)diagnóstico

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['max']
pre_cut_text = ['máxima', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_diagnostico_covid19']
pre_prepared_query = 'devolva a idade no município de São Paulo diagnóstico'

NLIDB Interface Properties

group_ranking_to_df:
idade:casos.idade:NT;paulo:casos.nome_munic:VT;diagnóstico:casos.diagnostico_covid19:NT;

nlidb_interface_sql = 'SELECT casos.diagnostico_covid19 as casos_diagnostico_covid19, casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['max']

pos_glamorise_sql = 'SELECT casos_diagnostico_covid19, max(casos_idade) as max_casos_idade FROM NLIDB_result_set GROUP BY casos_diagnostico_covid19 ORDER BY casos_diagnostico_covid19 LIMIT 100'

Results

	casos_diagnostico_covid19	max_casos_idade
0	CONFIRMADO	120.0
1	DESCARTADO	121.0

Figura A.40: *Resultado simplificado do processamento da consulta "Devolva a máxima idade no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

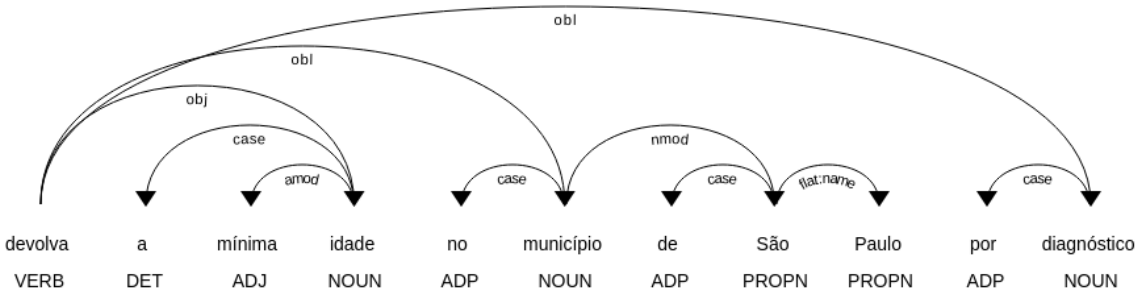


Figura A.41: *Diagrama de dependências da consulta "Devolva a mínima idade no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

Natural Language Query: devolva a mínima idade no município de São Paulo por diagnóstico**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']
[9, 'diagnóstico', 'ADJ', 7, 'São', 'amod']

Parse Tree:

(217)ROOT
  (218)devolva
    (219)idade
      (220)a
    (221)município
      (222)no
      (223)São
        (224)de
        (225)Paulo
        (226)diagnóstico

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['min']
pre_cut_text = ['mínima', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_diagnostico_covid19']
pre_prepared_query = 'devolva a idade no município de São Paulo diagnóstico'

NLIDB Interface Properties

group_ranking_to_df:
idade:casos.idade:NT;paulo:casos.nome_munic:VT;diagnóstico:casos.diagnostico_covid19:NT;

nlidb_interface_sql = 'SELECT casos.diagnostico_covid19 as casos_diagnostico_covid19, casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['min']

pos_glamorise_sql = 'SELECT casos_diagnostico_covid19, min(casos_idade) as min_casos_idade FROM NLIDB_result_set GROUP BY casos_diagnostico_covid19 ORDER BY casos_diagnostico_covid19 LIMIT 100'

Results

	casos_diagnostico_covid19	min_casos_idade
0	CONFIRMADO	0.0
1	DESCARTADO	0.0

Figura A.42: *Resultado simplificado do processamento da consulta "Devolva a mínima idade no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

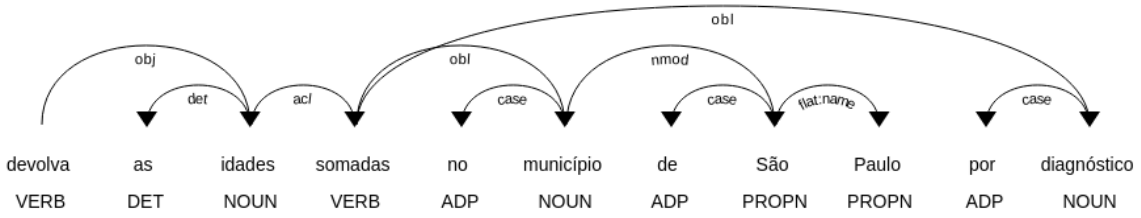


Figura A.43: *Diagrama de dependências da consulta "Devolva as idades somadas no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

Natural Language Query: devolva as idades somadas no município de São Paulo por diagnóstico**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'as', 'DET', 3, 'idades', 'det']
[3, 'idades', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']
[9, 'diagnóstico', 'ADJ', 7, 'São', 'amod']

Parse Tree:

(228)ROOT
  (229)devolva
    (230)idades
      (231)as
    (232)município
      (233)no
      (234)São
        (235)de
        (236)Paulo
        (237)diagnóstico

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['sum']
pre_cut_text = ['somadas', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_diagnostico_covid19']
pre_prepared_query = 'devolva as idades no município de São Paulo diagnóstico'

NLIDB Interface Properties

group_ranking_to_df:
idades:casos.idade:NT;paulo:casos.nome_munic:VT;diagnóstico:casos.diagnostico_covid19:NT;

nlidb_interface_sql = 'SELECT casos.diagnostico_covid19 as casos_diagnostico_covid19, casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['sum']

pos_glamorise_sql = 'SELECT casos_diagnostico_covid19, sum(casos_idade) as sum_casos_idade FROM NLIDB_result_set GROUP BY casos_diagnostico_covid19 ORDER BY casos_diagnostico_covid19 LIMIT 100'

Results

	casos_diagnostico_covid19	sum_casos_idade
0	CONFIRMADO	19814.0
1	DESCARTADO	18979.0

Figura A.44: *Resultado simplificado do processamento da consulta "Devolva as idades somadas no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

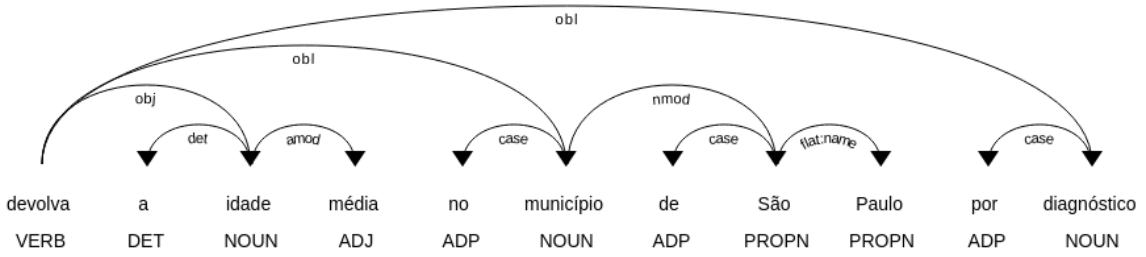


Figura A.45: *Diagrama de dependências da consulta "Devolva a idade média no município de São Paulo por diagnóstico", mencionada na seção 5.4.6*

Natural Language Query: devolva a idade média no município de São Paulo por diagnóstico**

Tree Table:

[1, 'devolva', 'VERB', 0, 'ROOT', 'root']
[2, 'a', 'DET', 3, 'idade', 'det']
[3, 'idade', 'NOUN', 1, 'devolva', 'obj']
[4, 'no', 'ADP', 5, 'município', 'case']
[5, 'município', 'NOUN', 1, 'devolva', 'obl']
[6, 'de', 'ADP', 7, 'São', 'case']
[7, 'São', 'PROPN', 5, 'município', 'nmod']
[8, 'Paulo', 'PROPN', 7, 'São', 'flat:name']
[9, 'diagnóstico', 'ADJ', 7, 'São', 'amod']

Parse Tree:

(239)ROOT
  (240)devolva
    (241)idade
      (242)a
    (243)município
      (244)no
      (245)São
        (246)de
        (247)Paulo
        (248)diagnóstico

GLAMORISE Internal Properties

Preprocessor Properties
pre_aggregation_fields = ['casos_idade']
pre_aggregation_functions = ['avg']
pre_cut_text = ['média', 'por']
pre_group_by = True
pre_group_by_fields = ['casos_diagnostico_covid19']
pre_prepared_query = 'devolva a idade no município de São Paulo diagnóstico'

NLIDB Interface Properties

group_ranking_to_df:
idade:casos.idade:NT;paulo:casos.nome_munic:VT;diagnóstico:casos.diagnostico_covid19:NT;

nlidb_interface_sql = 'SELECT casos.diagnostico_covid19 as casos_diagnostico_covid19, casos.idade as casos_idade FROM casos WHERE casos.nome_munic LIKE "%Paulo%"'

Postprocessor Properties
pos_aggregation_fields = ['casos_idade']
pos_aggregation_functions = ['avg']

pos_glamorise_sql = 'SELECT casos_diagnostico_covid19, avg(casos_idade) as avg_casos_idade FROM NLIDB_result_set GROUP BY casos_diagnostico_covid19 ORDER BY casos_diagnostico_covid19 LIMIT 100'

Results

	casos_diagnostico_covid19	avg_casos_idade
0	CONFIRMADO	40.519427
1	DESCARTADO	41.529540

Figura A.46: *Resultado simplificado do processamento da consulta "Devolva a idade média no município de São Paulo por diagnóstico", 5.4.6*

Referências

[Barbosa *et al.* 2017] J Barbosa *et al.* "Introdução ao processamento de linguagem natural usando python". Em: *III Escola Regional de Informatica do Piauí* 1 (2017), pgs. 336–360 (citado nas pgs. 4–7).

[Chamberlin 2009] Don Chamberlin. "Sql". Em: *Encyclopedia of Database Systems*. Ed. por LING LIU e M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pgs. 2753–2760. isbn: 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_1091. url: https://doi.org/10.1007/978-0-387-39940-9_1091 (citado na pg. 9).

[Contributors 2022a] Universal Dependencies Contributors. *Universal Dependencies*. 2022. url: https://universaldependencies.org/docs/u/dep/ (acesso em 28/08/2022) (citado nas pgs. 25, 44).

[Contributors 2022b] Universal Dependencies Contributors. *Universal POS tags*. 2022. url: https://universaldependencies.org/u/pos/ (acesso em 28/08/2022) (citado nas pgs. 25, 44).

[Devlin *et al.* 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee e Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. doi: 10.48550/ARXIV.1810.04805. url: https://arxiv.org/abs/1810.04805 (citado na pg. 2).

[Elmasri e Navathe 2015] Ramez Elmasri e Shamkant B. Navathe. *Fundamentals of Database Systems*. 7th. Pearson, 2015. isbn: 0133970779 (citado nas pgs. 2, 7, 9, 11).

[Explosion 2022a] Explosion. *Industrial-Strength Natural Language Processing*. 2022. url: https://spacy.io/ (acesso em 07/05/2022) (citado na pg. 20).

[Explosion 2022b] Explosion. *Using displaCy in Jupyter notebooks*. 2022. url: https://spacy.io/usage/visualizers#jupyter (acesso em 28/08/2022) (citado na pg. 44).

[Indurkhya e Damerau 2010] N. Indurkhya e F.J. Damerau. *Handbook of Natural Language Processing*. Chapman & Hall/CRC machine learning & pattern recognition series. CRC Press, 2010. isbn: 9781420085938. url: https://books.google.com.br/books?id=nK-QYHZ0-%5C_gC (citado nas pgs. 4, 5, 7).

[José *et al.* 2022] Marcos Menon José, Marcelo Archanjo José, Denis Deratani Mauá e Fábio Gagliardi Cozman. "Integrating question answering and text-to-sql in portuguese". Em: *Computational Processing of the Portuguese Language*. Ed. por Vládia Pinheiro *et al.* Cham: Springer International Publishing, 2022, pgs. 278–287. isbn: 978-3-030-98305-5 (citado na pg. 2).

[Li e Jagadish 2014] Fei Li e H. V. Jagadish. "Constructing an interactive natural language interface for relational databases". Em: *Proc. VLDB Endow.* 8.1 (set. de 2014), pgs. 73–84. issn: 2150-8097. doi: 10.14778/2735461.2735468. url: https://doi.org/10.14778/2735461.2735468 (citado nas pgs. v, 1, 2, 13, 15).

[Liao *et al.* 2020] Zhibin Liao *et al.* "Medical data inquiry using a question answering model". Em: *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. 2020, pgs. 1490–1493. doi: 10.1109/ISBI45749.2020.9098531 (citado na pg. 2).

[Moura 1996] Raimundo Santos Moura. "Sos - Um sistema orto-sintático para a língua portuguesa". Diss. de mestr. Pernambuco, Brasil: Universidade Federal de Pernambuco, 1996 (citado na pg. 5).

[Novello 2021] Alexandre Ferreira Novello. "A Novel Solution to Empower Natural Language Interfaces to Databases (NLIDB) to Handle Aggregations". Diss. de mestr. Rio de Janeiro, Brasil: Pontifícia Universidade Católica do Rio de Janeiro, abr. de 2021 (citado nas pgs. v, 1, 2, 15, 16, 21, 42).

[Oracle 2022] Oracle. *What is MySQL?* 2022. url: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html (acesso em 25/08/2022) (citado na pg. 23).

[Project 2022] NLTK Project. *Natural Language Toolkit*. 2022. url: https://www.nltk.org/ (acesso em 07/05/2022) (citado na pg. 21).

[SEADE 2022a] Fundação SEADE. *Repositório de dados sobre casos e óbitos decorrentes do COVID-19 nos municípios do Estado de São Paulo e sobre leitos e internações por Departamento Regional de Saúde*. 2022. url: https://github.com/seade-R/dados-covid-sp (acesso em 07/05/2022) (citado nas pgs. 3, 22).

[SEADE 2022b] Fundação SEADE. *Repositório de dados sobre casos e óbitos decorrentes do COVID-19 nos municípios do Estado de São Paulo e sobre leitos e internações por Departamento Regional de Saúde*. 2022. url: https://github.com/seade-R/dados-covid-sp#casos-%C3%B3bitos-por-ra%C3%A7acor-e-munic%C3%ADpio (acesso em 25/08/2022) (citado na pg. 22).

[Thanaki 2017] J. Thanaki. *Python Natural Language Processing*. Packt Publishing, 2017. isbn: 9781787285521. url: https://books.google.com.br/books?id=IedDDwAAQBAJ (citado na pg. 4).