Anton Bulle Labate

Aprimoramento de Question Answering por Texto em Linguagem Natural para SQL

São Paulo, SP

2023

Anton Bulle Labate

Aprimoramento de Question Answering por Texto em Linguagem Natural para SQL

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Universidade de São Paulo – USP

Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Prof. Dr. Fabio Cozman

São Paulo, SP

2023

Catalogação-na-publicação

Labate, Anton
Aprimoramento de Question Answering por Texto em Linguagem Natural para SQL / A. Labate -- São Paulo, 2023.
53 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Sintaxe e semântica da linguagem natural 2.Redes neurais 3.Aprendizagem profunda 4.Sistemas de questões e respostas I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Este trabalho é em homenagem aos meus pais,
que me impulsionam e inspiram a ser melhor, e aos professores que me formaram,
e me ensinam até hoje.

Agradecimentos

Agradeço aos meus pais, por me apoiarem nas minhas empreitadas, e ao meu orientador, pelas conversas e apoio. Também agradeço a vários professores que tive, os melhores que alguém poderia pedir, cujos ensinamentos e valores carrego comigo por onde vou e trazem para mim os meus sonhos.

Resumo

Na era de Big Data, em que vivemos, é intensa a produção e consumo, para análises precisas, de uma grande volume de dados. Por isso, também com o apoio e desenvolvimento das arquiteturas transformers, cada vez maiores e com resultados melhores, tem sido grande o interesse recentemente no desenvolvimento de mecanismos de tradução de linguagem natural para linguagens estruturadas e de acesso a banco de dados, como SQL, com o intuito de tornar tais bancos de dados mais acessíveis para pessoas sem conhecimento técnico. Porém, assim como em tarefas que também transformers, por vezes os modelos não interpretam corretamente as relações entre termos da frase, responsáveis por transmitir, pela forma como se dão e pelos termos que são envolvidos, a intenção do usuário. Assim, sem uma interpetação precisa e geral sobre as relações que verdadeiramente moldam à expressão da intenção do usuário, os modelos não conseguem traduzi-las para suas correspondentes representações na saída, produzindo um resultado distinto do pretendido. Para que os modelos compreendam e assimilem corretamente as interações descritas an frase de entrada entre termos, e, assim, transfiram-nas também à saída, respondendo conforme a intenção do usuário, temos uma infusão de representação sintática e semântico sobre a frase com a própria frase na entrada do modelo. A decomposição da frase em sua sintaxe e forma semântica é feita por dois parsers independentes, cujas saídas são linearizadas para concatenação com a frase original para formar a entrada para o modelo. Para pôr à prova o processo, treinamos uma mesma arquitetura, usando dois modelos de linguagem diferentes (ambos do tipo encoder-decoder), para cada tipo de informação apresentada (incluindo nenhuma informação, para controle e benchmark das pontuações). Verificamos que, tanto para 32 épocas quanto para 128 épocas, os modelos que tiveram apresentação de conhecimento, tanto sintático quanto semântico, tiveram pontuações maiores do que os sem informação. Portanto, os resultados mostram que a apresentação de informação sobre a frase é relevante para os modelos. Além disso, os modelos com informação tiveram, com apenas 32 épocas, pontuações próximas às que tiveram com 128 épocas de treinamento, indicando que um treinamento mais eficiente.

Palavras-chave: Transformers. Apresentação de conhecimento. Tradução de linguagem natural para SQL.

Abstract

In the era of Big Data, in which we live, the production and consumption, for precise analysis, of a large volume of data is intense. Therefore, also with the support and development of transformer architectures, increasingly larger and with better results, there has been great interest recently in the development of natural language translation mechanisms for structured and database access languages, such as SQL, with the aim of making such databases more accessible to people without technical knowledge. However, just like in tasks that also involve transformers, sometimes the models do not correctly interpret the relationships between terms in the sentence, responsible for transmitting, through the way they occur and the terms that are involved, the user's intention. Thus, without a precise and general interpretation of the relationships that truly shape the expression of the user's intention, the models are unable to translate them into their corresponding representations in the output, producing a result different from that intended. In order for the models to correctly understand and assimilate the interactions described in the input sentence between terms, and thus also transfer them to the output, responding according to the user's intention, we have an infusion of syntactic and semantic representation over the sentence with its own phrase in the model input. The decomposition of the sentence into its syntax and semantic form is done by two independent parsers, whose outputs are linearized for concatenation with the original sentence to form the input to the model. To test the process, we trained the same architecture, using two different language models (both encoder-decoder types), for each type of information presented (including no information, to control and benchmark scores). We found that, for both 32 epochs and 128 epochs, the models that presented knowledge, both syntactic and semantic, had higher scores than those without information. Therefore, the results show that the presentation of information about the sentence is relevant to the models. Furthermore, the models with information had, with just 32 epochs, scores close to those they had with 128 epochs of training, indicating more efficient training.

Keywords: Transformers. Knowledge presentation. Natural language to SQL translation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

NL2SQL	Natural language to SQL (Linguagem natural para SQL)
RESDSQL	Ranking enhanced encoder skeleton aware decoder to SQL
SQL	Structured Query Language (Linguagem de consulta estruturada)
QA	Question answering
seq2seq	Sequence to sequenc

Sumário

1 Introdução

A extração de informações de bancos de dados relacionais, com o crescente armazenamento e uso de dados para análise, tornou-se frequente e imprescindível para geração de análises precisas para empresas. As consultas, porém, aos bancos de dados relacionais são tipicamente feitas por linguagens como SQL, o que torna esse acesso muitas vezes obscuro para quem não tem conhecimento técnico. Os recentes avanços na área de deep learning renovaram o interesse na investigação de interfaces de linguagem natural para acessar bancos de dados (WANG et al., 2020; IACOB et al., 2020), numa resposta para esse problema. Nessa abordagem *Question Answering* (QA), modelos de linguagem do tipo seq2seq (Sequence to Sequence), transformers, são usados para a tradução de textos linguagem natural para SQL. Todavia, a ambiguidade na questão ou até mesmo a estrutura de formulação da pergunta (interações entre palavras) impõem muitas vezes um obstáculo para que o modelo compreenda corretamente a consulta e gere a saída que corresponda a intenção real do usuário, assim como para tarefas que tambem envolvem a arquitetura transformer e são sensíveis à frase de entrada do usuario, isto é, por pequenas variações na entrada, grandes alterações saõ provocadas na saída. Assim, nessa monografia tratamos uma alternativa para isso

1.1 Motivação

A arquitetura de rede neural transformer tem apresentado grande habilidade em tarefas de processamento de linguagem natural, por exemplo para *Question answering*, sumarização e tradução. Porém, apesar disso, os modelos ainda têm problemas para interpretar estruturas frasais (LEIVADA; MURPHY; MARCUS, 2022). Esse problema pode ser observado em tarefas sensíveis à frase de entrada do usuário, isto é, tarefas nas quais é vital a correta e completa assimilação da frase do usuário para a geração da saída pedida, quando o modelo, por não interpretar assertivamente e corretamente algum ponto da frase (relação entre termos ou natureza de uma relação), gera uma resposta que não traduz a intenção da frase. Por exemplo, na tradução de linguagem natural, os adjetivos na língua do texto fonte modificam apenas os substantivos a que se associam, e esta interação tem que ser respeitada na saída na língua alvo. Na tradução de linguagem natural para SQL, essa tradução incorreta da intenção da frase do usuário para uma consulta em SQL é visível nas frases abaixo, extraídas da base de dados Spider (YU et al., 2019), nas quais apresentamos a frase de entrada, feita pelo usuário, e a consulta gerada pelo modelo:

```
Quais são os ids e fabricantes de todos os fabricantes de carros que
produzem pelo menos 2 modelos e fazem mais de 3 carros?
```

```
Resposta SQL: select car_makers.Id, car_makers.Maker FROM car_makers
              JOIN model_list
              GROUP BY car_makers.Id HAVING Count(*) >= 'terminal' INTERSECT
              SELECT car_makers.Id, car_makers.Maker FROM car_makers
              JOIN model_list
              GROUP BY car_makers.Id HAVING Count(*) >= 'terminal'
```

Nessa resposta, o modelo repete a primeira restrição feita (sobre o fabricante ter mais de 2 modelos), após o intersect, no lugar de adicionar a restrição sobre fabricar mais de três carros.

```
Quais são os modelos mais leves do que 3500 mas que não forma feitos
pela montadora X?

Resposta SQL: model_list.Model FROM model_list JOIN car_names JOIN car_makers
              WHERE car_makers.FullName = 'X'
              OR cars_data.Weight < '3500'

Resposta correta: model_list.Model FROM model_list JOIN car_names JOIN car_makers
                  WHERE  not car_makers.FullName = 'X'
                  and cars_data.Weight < '3500'
```

Aqui, o modelo não traduz corretamente a relação aditiva de negação ("mas que não") para um "not"(para negar o nome da montadora) seguido por "and"(ter também a restrição sobre peso), mas para um "or"sem "not".

Na abordagem por aprendizado de máquina para a questão de *Natural Language to SQL*, o *benchmark* para a tarefa é a base de dados Spider. No Spider, os modelos estado-da-arte para a tradução de texto em Inglês predominantemente recorrem à arquitetura GPT para auxílio (GAO et al., 2023). Entre os modelos que se fundamentam em arquiteturas públicas, como o modelo de linguagem T5, está o RESDSQL (LI et al., 2023), que será a arquitetura do trabalho. Em Português para SQL, o estado-da-arte é o RAT-SQL+GAP (JOSÉ; COZMAN, 2021). Nesse modelo, o RAT-SQL (Relation aware) serve ao modelo para treiná-lo a identificar as entidades (associar) do *schema* do banco de dados às suas respectivas referências na consulta em linguagem natural. O GAP no modelo gera dados (sintéticos) para aumentar o dataset e ter um pré-treinamento numa base maior. O modelo de linguagem com essas configurações que é refinado para a tradução para SQL no treinamento é a versão multilíngual do Bart (mBart). Para treinamento com frases em

Português e Inglês e inferência tambem com frases em Português e Inglês, o modelo tem 0,630 na métrica exact set match em testes gerais. No framework RESDSQL, é feito um preprocessamento sobre a frase de entrada do usuário e o *schema* (relações de tabelas e respectivas colunas) da base de dados, para que apenas os itens da base de dados (tabelas e respectivas colunas) mais próximos (relacionados) da pergunta feita pelo usuário sejam apresentados ao transformer. O decodificador, por sua vez, não gera a partir da saída do codificador diretamente a consulta SQL, mas antes gera o esqueleto da consulta em SQL (a estrutura), para produzir então a correspondente consulta SQL. No nosso modelo, faremos a tradução para Português e também apresentamos na entrada as informações sintáticas e semânticas sobre a frase.

1.2 Objetivos

Os resultados em trabalhos anteriores na área de Natural language to SQL foram visíveis, no modo geral, porém não satisfazem toda pergunta que um usuário típico tem interesse. Particularmente, por testes feitos com modelos próximos ao estado da arte, vemos que o modelo muitas vezes não sabe gerar a relação que traduz a intenção do usuário entre itens do banco de dados na sua consulta de saída. O problema tem relação com o próprio transformer (ou large language model), o qual é treinado com grandes quantidades de dados, sem informação sobre a estrutura, para que possa fazer a tradução para a saída. Para que o modelo se aproxime da real intenção do usuário e, assim, gere a saída pretendida, é preciso que interprete corretamente as relações expressas por termos da frase e a natureza delas. A proposta do trabalho é aprimorar a "consciência"linguistica do modelo das interações da frase, para garantir que as perguntas feitas pelo usuário, independentemente do nível, tenham a resposta correta para o usuário, com as relações entre os itens em conssonância com as relações vistas na frase do usuário, algo imprescindível numa linguagem estruturada como SQL. Assim, apresentamos no treinamento, por uma infusão de conhecimento, as formas sintáticas e semânticas das frases também na entrada.

1.3 Justificativa

As informações relevantes para a operação de uma empresa, vitais para a análise por parte dos seus funcionários, muitas vezes são armazenadas nas bases de dados relacionais, que tem interações tipicamente feitas por meio de linguagens como SQL e SPARQL. Porém, o uso dessas linguagens de consulta para pessoas que nao tem *expertise* técnica na área é muitas vezes um obstáculo ao uso dessas informações, o que geralmente as vincula ao serviço de pessoas que saibam interagir por essas linguagens com a base de dados. Isso torna o processo de resgate de respostas para as consultas do usuário de maneira geral um processo lento e inconveniente.

Por isso, tem havido intenso interesse por técnicas que facilitem o acesso a essas bases (OZCAN et al., 2020; KIM et al., 2020; AFFOLTER; STOCKINGER; BERNSTEIN, 2019). Particularmente, com as técnicas de deep learning recentemente, os modelos atuais permitem que a interpretação de perguntas (requisições) mais complexas do usuário, com união de tabelas e consultas (*queries*) aninhadas. Porém, esses modelos ainda tem problemas com algumas perguntas, por mais fáceis que sejam na classificação da base. Por testes preliminares usando o modelo atual, próximo ao topo do ranking para NL2SQL, para uma base de dados qualquer, foi possível ver que, apesar de responder corretamente algumas perguntas, não são todas as perguntas, por mais usuais, que têm a resposta certa pelo modelo. Assim, a resposta atual ainda o vincula à necessidade de uma avaliação por alguém que saiba SQL para gerar a *query* final. A proposta da pesquisa é tentar tornar o modelo próprio para uso por alguém que não saiba SQL, ao mesmo tempo que tenha respostas para muitas perguntas usuais que faria, para que qualquer um tenha acesso aos bancos de dados. Nisso, poderíamos aumentar a precisão do modelo para que também atenda pessoas que tenham consultas mais robustas. Para isso, o modelo teria uma forma que avalia tanto as informações sintáticas na entrada, como a pergunta em si na linguagem natural (treinamento usual dos transformers). Para tarefas como análise de sentimento (BAI et al., 2021a), ou classificação (ZANZOTTO et al., 2020), tal apresentação de conhecimento, embora feita de variadas formas, teve resultados promissores.

O modelo que atende questões usuais, sem que a pessoa que o usa saiba a linguagem, permitiria que muitos usuários se valessem da riqueza dos vários bancos de dados, que servem para armazenar informações para muitas empresas. Assim, analistas, gerentes ou consultores teriam maior facilidade e agilidade para obter relatórios, sem que tivessem que recorrer a outros.

1.4 Organização do Trabalho

A seguir explicitamos os temas que tratamos por capítulo. No capítulo 2, ilustramos as diferentes abordagens possíveis para a tarefa de NL2SQL (tradução de linguagem natural para SQL), explorando com mais detalhes a que usamos no trabalho, com inclusive uma dissecação da arquitetura para a tarefa. Ainda nesse capítulo, tratamos o tema da infusão de conhecimento. Primeiro, mostramos um panorama de artigos que também se valeram de representações sobre a frase (morfológicas principalmente), de várias maneiras, para que, com uma supervisão sobre as interações da frase, os modelos tenham maior consciência sobre a forma linguística da frase, proporcionando mais precisão às tarefas. Ainda, apresentamos a nossa forma de infusão de conhecimento e os tipos de informação com que trabalhamos.

No capítulo 3, referimo-nos aos processos para implementação do projeto. No-

minalmente, descrevemos as etapas do desenvolvimento prático, conteúdo dos próximos capítulos.

O capítulo 4 trata dos aspectos práticos e imprescindíveis dos requisitos para a apresentação de conhecimento ao modelo tradutor de linguagem natural para SQL. Para que possamos treinar o modelo, precisamos definir um *dataset* de perguntas e respectivas consultas SQL. Também estudamos os parsers que adotamos para a decomposição da frase em sua estrutura sintática e semântica, assim como seus formatos para saída.

No capítulo 5, tratamos a integração dos parsers ao treinamento do modelo. Assim, especificamos o tratamento (pré-processamento) feito sobre a saída de cada parser para integrá-la à frase da base de dados cuja forma, sintática ou semântica, representa. O treinamento é descrito, assim como método para avaliação dos modelos e validação.

No capítulo 6, temos a ponderação sobre os resultados do trabalho e contribuições. Ainda, apontamos trabalhos de continuidade.

2 Aspectos Conceituais

A tradução de linguagem natural - SQL, NL2SQL, faz a passagem de uma consulta em linguagem natural para sua respectiva consulta (*query*) em SQL. Abaixo, mostramos qual seria a saída para um modelo que faça essa operação de NL2SQL para a pergunta "Qual é o número de carros com mais que 4 cilindros?".

```
    Select count (*)
From cars_data
Where cilindros > 4
```

A tradução de Natural Language - SQL (NL2SQL) em geral é abordada por regras ou por aprendizado de máquina, na qual se classifica a arquitetura que usamos. A seguir tratamos essas abordagens, em particular a arquitetura para o nosso modelo. Nas próximas partes, exploramos os tipos de informação que apresentamos ao modelo, para que tenha interpretação das relações entre os termos da frase de entrada e a forma como se dão e apresentações de outros tipos de informação na literatura para outras tarefas.

2.1 Abordagem por regras

A abordagem por regras tenta reconhecer os termos na frase como suas respectivas formas no banco de dados e suas relações baseados em formas de apresentação dos dados internos (taxonomia, ontologia). O sistema Nalir (LI; JAGADISH, 2014) é um sistema que se vale disso, representando a query na entrada em uma árvore resultante de um *parsing* morfológico, o que a aproxima da forma linguística da questão. Por meio dessa árvore, interpretável pelo usuario e pelo modelo, o sistema faz as associações entre itens dela e nomes no banco de dados e, quando não sabe fazer isso (incerto sobre as associações), pergunta ao usuário qual das árvores que gera é a resposta para a pergunta. Assim que tem a árvore e suas associações feitas de forma válida (dentro das restrições de vocabulário de SQL, isto é, são mapeamentos válidos), é produzida a saída em SQL para o usuário.

O ATHENA (SAHA et al., 2016), por sua vez, mapeia partes da consulta em linguagem natural para conceitos e relacionamentos em uma ontologia que captura a semântica da base de dados. O ATHENA usa uma linguagem intermediária de consulta antes de traduzir a consulta de entrada para a saída.

Tais sistemas usam conhecimentos de taxonomias e ontologias, o que facilita incorporar conhecimento de domínio (e torna isso obrigatório para inferência). Porém, são muito sensíveis a variações e paráfrases da frase de entrada do usuário, uma vez que as

palavras do usuário são usadas para associação com algum termo da base de dados, e ainda precisam de interação do usuário (por exemplo, para definir regras).

2.2 Abordagem baseada em aprendizado de máquina

Na abordagem baseada em aprendizado de máquina, as redes realizam treinamento por pares de perguntas em linguagem natural e consultas em SQL (aprendizado supervisionado). Um exemplo é o sistema RAT-SQL+GAP e o RESDSQL, que usamos no trabalho. As abordagens por machine learning se mostram flexíveis nos resultados, permitem mais operações sobre as bases de dados (uniões e consultas que tem mais requisitos, assim como variações na forma para se referir a um item).

Para treinamento desses modelos temos algumas bases de dados. O Spider, que foi a base de dados usada para treinar o modelo que usaremos, é uma das mais novas propostas de base para NL2SQL e se apresenta como uma base que tem *queries* com vários níveis de SQL, não apenas simples, para vários tipos de aplicação. Tipicamente, os primeiros sistemas tratavam a tarefa como uma de *parsing* ou tradução mesmo, dado uma base de dados e a pergunta, concatenavam as tabelas e suas respectivas colunas (conjunto a que nos referimos como *schema*) da base com a pergunta (em ordem aleatória, ou padrão) e isso era a entrada para o modelo *sequence-to-sequence* (seq2seq), que por sua vez gerava a correspondente query SQL.

Alguns exemplos mais de sistemas baseados nessa abordagem são o Picard (SCHOLAK; SCHUCHER; BAHDANAU, 2021) e TypeSQL (YU et al., 2018). No Picard, é feita uma restrição sobre as consultas SQL feitas pelo modelo de linguaem para apenas consultas válidas em SQL. Assim, a consulta é correta tanto gramaticalmente (com palavras válidas), quanto na sua estrutura, valendo para trasnformers pré-treinados e sem requerer uma vasta geração de saídas para sua análise. O TypeSQL usa grafos da frase ou conteúdos de tabela para o modelo compreender termos e números na questão. A seguir tratamos do framework que usamos no nosso modelo em particular.

2.2.1 Framework RESDSQL

O framework RESDSQL é baseado na arquitetura sequence-to-sequence (seq2seq) (fundamenta-se no uso de uma arquitetura do tipo sequence - to - sequence para a tarefa de traduzir linguagem natural para SQL, particularmente um transformer encoder-decoder). Tipicamente, nessa tarefa, dada uma pergunta em linguagem natural, é concatenada a ela o schema da base de dados (tabelas e colunas), e é pedido a um modelo seq2seq (modelo de linguagem pré-treinado), como Bart ou T5, para gerar a consulta SQL associada. Na figura 1, temos um exemplo para uma base de dados do Spider.

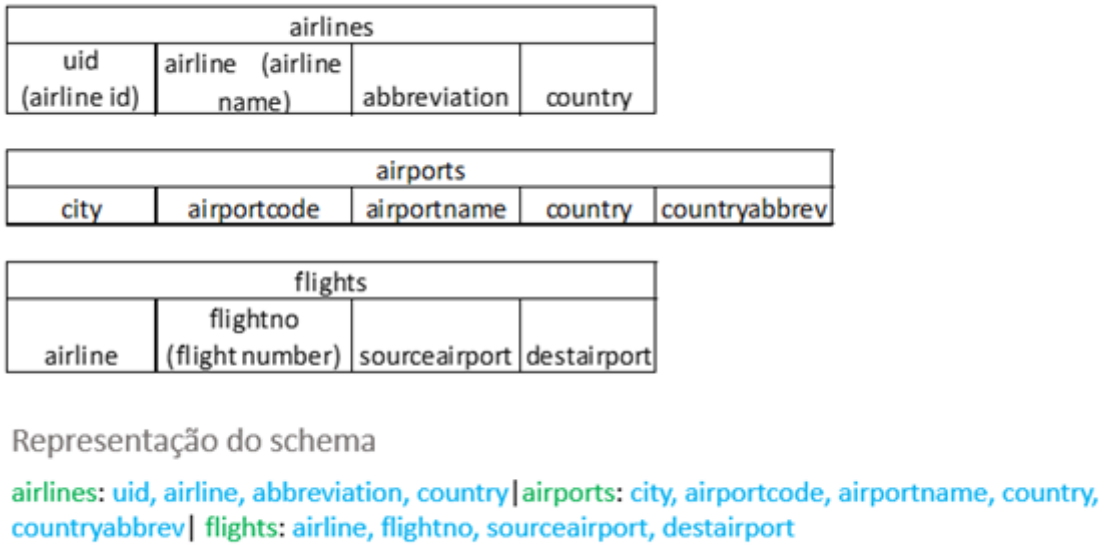

Figura 1 – Base de dados de aeroportos do Spider

No schema, inserimos sequencialmente as tabelas e suas respectivas colunas, com um separador "entre tabelas. Assim, para a pergunta "Quais são os voos saindo da cidade "York"?", teríamos para a entrada do modelo:

```
"Quais são os voos saindo da cidade "York"? | airlines: uid, airline,
abbreviation, country|airports:city, airportcode,airportname, country,
countryabbrev| flights: airline, flightno, sourceairport, destairport"
```

No framework RESDSQL, há uma separação da tradução da pergunta para SQL entre as sub-tarefas de identificação dos itens do schema mencionados na pergunta, e que devem portanto integrar a query resultante, e a geração do esqueleto (estrutura) de SQL respectivo, para que possamos, ao final das etapas, preencher essa estrutura SQL gerada com os itens do schema necessários para a formação da query SQL resultante.

2.2.1.1 Ranking-Enhanced Encoder (Encoder com aprimoramento de ordenação)

Na primeira parte, elencamos todos os itens do schema, tabelas e colunas, de acordo com a sua probabilidade de observação, pela pergunta feita, de forma decrescente, para que, na concatenação, os itens mais relevantes tenham prioridade e apareçam antes de itens que tem menor probabilidade. Para isso, treinamos um cross-encoder, um modelo baseado no BERT. O input, X, para o cross-encoder é feito pela concatenação da pergunta original ao schema da base de dados, com cada tabela, com suas respectivas colunas, separada das outras por um separador ". Na figura 2 retratamos a operação do cross-encoder.

Figura 2 – Alimentação do cross-encoder com a entrada X

Os itens do schema, para se aproximarem das palvras na pergunta, são traduzidos para sua forma semântica (e.g., airport.id para airport id). Na figura 2, na parte do schema do banco de dados, trocamos uid, o nome da coluna referente ao id da linha aérea, por airline id, o nome semântico. Alimentamos o cross-encoder, um encoder RoBERTa (LIU et al., 2019), uma versão do BERT (DEVLIN et al., 2019), com X, com o cuidado para que, após a tokenização do input, reagrupemos os tokens referentes a um mesmo item do schema que foi separado pelo tokenizador, e assim não tenhamos um mapeamento de termo do schema para mais de um token (e.g., airport id para dois tokens separados, "airport"e "id"). É feita também a mistura da relação das colunas pertencentes à tabela com a embedding da própria tabela, para que a falta de menção do nome da tabela na pergunta não impeça o encoder de identificá-la.

A perda do cross-encoder é definida tanto pelas colunas quanto tabelas. Por as consultas em SQL envolverem usualmente poucas colunas e tabelas da base de dados, há grande grau de desbalanceamento entre casos postivos, de verificarmos a aparição de uma coluna ou tabela na frase, em relação aos casos negativos, o que poderia induzir um viés no treinamento se usássemos entropia cruzada (cross-entropy, tipicamente usada para classificação). Assim, a função de perda que usamos a é a perda focal:

Assim que tenhamos as probabilidades de observação dos itens do schema para a pergunta feita, filtramos os termos para que apresentemos ao modelo seq2seq apenas as colunas ou tabelas mais relevantes. Formalmente, concatenamos à pergunta as k1 tabelas mais relevantes, com as suas k2 colunas mais relevantes, em ordem descrescente de probabilidade. O k1 e k2 são hiperparâmetros, que se pequenos levam ao descarte de tabelas potencialmente referencidas na pergunta, e se grandes inserem tabelas irrelevantes, como ruído. A concatenação da pergunta a essa sequência de itens do schema e opcionalmente também com relações de chaves estrangeiras (que poderiam promover uma ajuda na geração de cláusulas do tipo JOIN ON) é a entrada do modelo seq2seq.

2.2.1.2 Decoder Consciente do esqueleto

Geralmente, as arquiteturas seq2seq para a tarefa de tradução de linguagem natural para SQL fazem a geração direta do SQL correspondente à requisição. Para reduzir a distância entre a consulta em SQL e a pergunta em linguagem natural, porém, a arquitetura RESDSQL gera primeiro o esqueleto da consulta em SQL para, na sequência, produzir a query em SQL. Para realizar essa decomposição sem a adição de mais módulos, é proposto um novo alvo para saída do decoder, na sua propriedade de gerar o i-ésimo token baseado não apenas no input, mas também nos tokens gerados (até o i-ésimo -1 token). Assim, o decoder primeiro decodifica o esqueleto do SQL correspondente à pergunta, para após isso gerar o SQL resultante, seja copiando alguma parte do esqueleto gerado por ele mesmo anteriormente, ou algum item da sua entrada. Para a pergunta "Quantos voos temos?", a respectiva consulta em SQL seria "select count (*) from flights". O decoder teria na sua saída:

```
select count ( _ ) from _ | select count ( * ) from flights
```

Para a extração do esqueleto do SQL, primeiro foi feita uma normalização das consultas em SQL, usando as mesmas convenções de escrita para cada uma delas, por exemplo palavras-chave em SQL ("where") em letras minúsculas e aspas simples no lugar de aspas duplas. Pela normalização das consultas em SQL, podemos extrair o esqueleto, que contém apenas as palavras reservadas em SQL e lacunas no lugar dos itens do schema.

Porém, não restringimos o decoder com a gramática do SQL. Para garantir a geração de consultas estruturalmente corretas em SQL (bem formadas válidas/executáveis em SQL), usamos um seletor guiado à execução de SQL, que realiza uma busca no feixe de geração (beam search) no processo de decoding e seleciona a primeira consulta SQL executável.

2.3 Apresentação de conhecimento e tipos de informação

Foram propostas algumas formas para realizar a apresentação de informações sobre a frase entrada, para várias tarefas, e vários tipos de informação. Punyakanok, Roth e Yih (2008) foi um dos primeiros a incorporar informação morfológica, para a tarefa de rotulação de papéis semânticos (sem transformer). As informações que apresentamos ao modelo de linguagem são de dois tipos: sintática e semântica. No artigo de (WILCOX et al., 2019), mostram que modelos de RNNs (redes neurais recorrentes) com supervisão da estrutura da frase tem resultados mais próximos ao humano do que modelos sem supervisão para a generalização e identificação de dependências gramaticais.

Árvores morfológicas também foram usadas no treinamento de modelos. Bai et al. (2021b) tem módulos de auto-atenção para cada relação morfológica na árvore, para

aplicação na análise de sentimento e inferência de linguagem natural. Para aumentar o resultado para a tradução, no (CURREY; HEAFIELD, 2019), árvores morfológicas também foram incorporadas à entrada de encoders, e, para modelos encoder-decoder, teve pré-treinamento na tarefa de parsing sintático para a tarefa de traducao de linguagem natural. O artigo (ZHANG et al., 2020b) usa uma rede para inserir relações morfológicas na operação de auto-atenção de modelos de linguagem (LM), em tarefas de comprrensão de leitura.

Também foram apresentadas outras relações, além das oriundas da morfologia da frase. No artigo (ZHANG et al., 2020a), é mostrado um grafo similar a Abstract Meaning Representation da entrada no treinamento de uma arquitetura encoder para várias tarefas de classificação. No artigo (WU; PENG; SMITH, 2021), também apoia-se no uso de informação semântica (particularmente a representação dm), para classificação por um encoder.

Nossa proposta é realizar a apresentação de dois tipos de informação sobre a frase para o modelo, sem que tenhamos que alterar estruturas internas do transformer, como fazem alguns artigos, mas apenas incorporando-a diretamente na entrada do modelo de linguagem. Assim, nossa proposta, ao contrário da maioria da literatura tangente à incorporação de informação sobre a frase, não requer um custo de inferência adicional ou retreinamento de modelos de linguagem, mas mostra-se uma abordagem prática que requer apenas o parsing da frase de entrada. Além disso, decompomos as frases em estruturas pouco usuais (todavia muito relevantes para a frase!) pela literatura, uma vez que a maioria se debruça sobre o uso de morfologia, quando requer informação sobre a frase. Os tipos de informação de que nos valemos são sintático e semântico.

2.3.1 Tipo sintático

A forma sintática da frase refere-se às funções sintáticas desempenhadas por cada termo e as relações entre essas funções. Logo, a forma sintática apresenta o sujeito da frase, o verbo raiz, os objetos direto e indireto, entre outras funções. Para capturar essas informações, recorremos a um parser de dependências, que as gera na forma de uma árvore, a qual linearizamos e inserimos junto com a respectiva frase na entrada. A árvore de dependências gerada pelo parser pode ser vista na figura 3.

Na figura, temos arcos que apontam do pai da relação para o filho, com o rótulo do arco descrevendo o tipo de relação que têm os termos envolvidos. Por exemplo, na relação entre "Tudo"e "vale", vemos que o arco aponta do verbo para "Tudo", indicando que o pai sintático do termo "Tudo"é o verbo "vale"(ou alternativamente "Tudo"é um filho de "vale"), e a relação que esses termos têm, pelo rótulo "nsubj"é de sujeito (o termo "Tudo"é o sujeito do seu pai sintático, "vale").

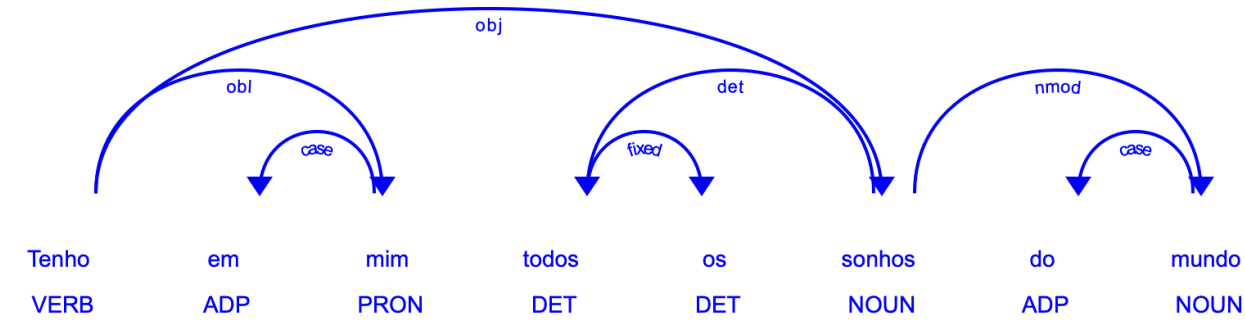

Figura 3 – A árvore de dependências da frase "Tenho em mim todos os sonhos do mundo".

Na figura 3, os arcos apontam do pai sintático para o filho sintático (dependente), com o rótulo do arco nomeando o tipo de relação que os dois termos envolvidos têm. Por exemplo, na relação entre "Tenho"e "sonhos", vemos que o arco aponta do verbo para "sonhos", indicando que o pai sintático do termo "sonhos"é o verbo "Tenho"(ou alternativamente "sonhos"é um filho de "Tenho"), e a relação que esses termos têm, pelo rótulo "obj", é de objeto direto (o termo "sonhos"é o objeto direto do seu pai sintático, "Tenho").

2.3.2 Tipo semântico

A representação semântica é representação apenas dos termos e relações que carregam o núcleo semântico da frase, o âmago da informação que ela transmite. Termos como determinantes, por exemplo, não são abrangidos, portanto (ao contrário da representação sintática). Para representarmos essa informação, recorremos à Abstract Meaning Representation (AMR) (BANARESCU et al., 2013).

AMR, ou Abstract Meaning Representation, é um grafo dirigido acíclico, cujos nós-folha correspondem a conceitos, e as arestas são associadas às relações entre eles. Os conceitos são palavras em Inglês ("boy", "pencil"), a que associamos uma variável na frase (nomeamos por "p"o primeiro lápis que aparece na frase, assim futuras referências a ele são referentes a "p), ou framesets de PropBank ("want-01") (PALMER; GILDEA; KINGSBURY, 2005; KINGSBURY; PALMER, 2002). A cada verbo, associamos um quadro, ou frameset, com os papéis (ou argumentos) semânticos esperados correspondentes a ele (que podem ocorrer ou não, a variar pelo caso de uso). Os papéis semânticos são numerados de 0 até 5 (se houver) e formalmente próprios a cada frameset, porém há muitos papéis comuns a vários framesets. Por exemplo, Arg0 geralmente é o argumento do verbo que mostra características de agente, enquanto Arg1 é frequentemente o paciente. Verbos polissêmicos podem ter mais de um frameset, quando as diferenças de significado são grandes o suficiente para requerer um grupo de papéis semânticos diferentes, um para cada frameset. Abaixo, por exemplo temos o frameset para o verbo "buy"(traduzido para

Português manualmente).

```
comprar
Arg0: comprador
Arg1: o que foi comprado
Arg2: vendedor, de quem se compra
Arg3: preço pago
Arg4: para que comprou
```

Não precisamos ter todos argumentos na frase. Por exemplo:

A companhia comprou um carregamento de máquinas de York

```
Arg0: companhia
Arg1: carregamento de máquinas
Arg2: York
```

Vemos ainda que para diferentes framesets, alguns papéis são comuns, como o Arg0 referente ao agente e Arg1 o paciente. O frameset de vender, por exemplo, mostra isso:

```
Vender
Arg0: vendedor
Arg1: o que é vendido
Arg2: comprador
Arg3: preço pago
Arg4: para que vendeu
```

Os grafos gerados podem ser linearizados usando a notação PENMAN. No trecho abaixo, vemos a linearização do grafo AMR correspndente à frase "everything is worthwhile if the soul is not small".

```
everything is worthwhile if the soul is not small
(w / worthwhile
:domain (e /everything)
:condition (s/ small
            :polarity -
            : domain (s2 /soul)))
```

Nessa representação, vemos a associação de "everything", termo nomeado por "e"(nome vem antes da barra "/"que precede o termo em si), como o domínio do termo "worthwhile"("w"). A condição para a relação anterior é vista pela negação do termo "small"(variável "s"), cujo domínio é "soul"("s2"). Também usamos um parser para gerar a correspondente árvore AMR da frase, a qual linearizamos e concatenamos à frase na entrada.

3 Método do trabalho

No início da pesquisa, após a incorporação e atualização de alguns parâmetros do sistema mRAT-SQL+GAP em Português para que pudesse gerar as consultas no ambiente de teste, foi feita a verificação do seu funcionamento para bases de dados fora do Spider, as bases a que tive acesso da Amazônia Azul. Porém, para que pudesse usá-las, tive que fazer o tratamento das tabelas de teste, pela renomeação de variáveis e uso do formato necessário. Após isso, algumas perguntas usuais sobre um grupo de tabelas testaram os resultados para tabelas não vistas antes, sequer na validação. Na sequência, no final do ano passado (2022), a investigação foi principalmente sobre as razões para o modelo não ter as respostas que queríamos para as consultas mais fáceis. A forma para isso foi por testes de perguntas relativas a algumas bases de dados (2) que tomamos aleatoriamente do próprio Spider e verificação da *query*. Foi visto que as consultas SQL de saída do modelo não traduziam as relações entre termos da frase, tanto quanto ao tipo de interação entre termos quanto aos termos propriamente que se relacionavam. Portanto, o modelo não interpretava corrretamente as interações entre os termos da frase, para gerar às relações respectivas em SQL. Para que o modelo tenha consciência dessas interações, apresentamos na entrada, concatenado à frase original, a decomposição sintática, ou semântica, tornando-as explicitas.

Para treinamento e validação dos resultados de um modelo, precisamos definir uma base de dados e um ter um benchmark, respectivamente. Assim, a primeira etapa do trabalho é tomar uma base de dados e um benchmark. Para isso, avaliamos alternativas e as comparamos. Para a tarefa da tradução de linguagem natural para SQL, a base de dados mais geral e completa e benchmark são um só, a base Spider.

A partir da base de dados escolhida, para que treinemos um modelo tradutor de Português especificamente para SQL, é imprescindível que traduzamos as perguntas dessa base de dados para Português (dado que a maioria das bases contém as perguntas em Inglês). Assim, teremos ao final desse processo uma base de dados com pares de perguntas em Português e suas consultas SQL correspondentes.

Para decompor as frases em suas estruturas sintática e semântica, precisamos de parsers. Portanto, estudamos os parsers para cada tipo de informação que queremos ter sobre a frase e como incorporar a sua saída à entrada do modelo de linguagem, com a frase. Para tanto, é possível que tenhamos um pós-processamento da saída dos parsers para termos a infusão desse conhecimento com a entrada, por exemplo a linearização da saída.

Com a base de dados traduzida para Português e com o conhecimento sintático,

ou semântico, incorporado a ela, podemos alimentar o nosso modelo com as perguntas da base. O modelo escolhido foi o RESDSQL, assim, para cada base, contendo um tipo (ou nenhum de informação) sobre as frases, treiná-lo-emos. Para verificar a relevância das informações, treinamos um modelo do RESDSQL sem informação alguma, como controle e referência para os novos modelos com apresentação de conhecimento.

4 Especificação de Requisitos

Embora o modelo apresente precisão razoável de modo geral para perguntas, ainda mostra por vezes problemas para responder alguns tipos de perguntas, por mais usuais que se apresentem, o que o afasta da intenção de servir como uma ferramenta que alivie o usuário do uso e geração por ele mesmo de linguagens de consulta. Por testes feitos no sistema para diagnosticar quais as principais razões para os erros na geração da consulta correta, foi visto que um problema usual é na interpretação das relações sintáticas entre termos expressas na linguagem natural, porque para várias amostras o modelo não as traduz para a relação associada na consulta. Para que o modelo a aprenda a traduzir as associações sintáticas na pergunta para a consulta SQL, no treinamento agregamos conhecimento sobre a frase. As novas informações que incorporamos, sintática e semântica, são produzidas por dois parsers, um de dependências e um que gera o AMR correspondente. Nas seções 4.2.1 e 4.2.2, apresentamos os parsers a que recorremos para cada tipo de informação.

Os pares de pergunta e consulta SQL com que treinamos o nosso modelo, e a cujas perguntas acrescentamos as informações são da base de dados Spider, que é o *benchmark* da tarefa de tradução de linguagem natural para SQL, ou seja, os resultados para validação nessa base de dados são uma referência para o desempenho. Na seção 4.1, estudamos esse *dataset* em particular e mostramos anteriores. Nas seções 4.2.1 e 4.2.2, exibimos os parsers que geram as informações sobre as frases

4.1 O dataset para tradução de linguagem natural para SQL

Os primeiros datasets para a tarefa de tradução de linguagem natural para SQL apresentavam apenas um domínio de aplicação, com poucas ou apenas uma base de dados, o que implicava no reuso da base de treinamento na validação, mascarando a capacidade de generalização dos modelos. Além disso, também eram restritos a uma pequena gama das formas lógicas de SQL, portanto tinham pouca variação nas estruturas das queries de saída. Assim, as mesmas estruturas alvo vistas no treinamento eram vistas na validação, o que permitia que modelos tivessem um desempenho razoável mesmo para requisições de um nível maior por memorização e replicação de padrão visto no treinamento. Os datasets que tinham mais bases de dados, como WikiSQL (ZHONG; XIONG; SOCHER, 2017), por sua vez, continham queries em SQL simples e tabelas únicas por base de dados. Portanto, algumas bases de dados, como ATIS (PRICE, 1990) e GeoQuery (ZELLE; MOONEY, 1996), apesar de terem estruturas lógicas mais complexas, por serem poucos padrões para varias perguntas (paráfrases), o modelo memorizava o tipo, ou formato de pergunta, a

Dataset	Perguntas	SQL	bases de dados	domínios	tabelas por base
GeoQuery	877	247	1	1	6
ATIS	5,280	947	1	1	32
WikiSQL	80,654	77,840	26,521	-	1
Spider	10,181	5,693	200	138	5.1

Tabela 1 – Comparação entre datasets de Linguagem Natural para SQL

uma saída em SQL, reutilizando também bases de treinamento nas tabelas envolvidas nas perguntas de validação, nao medinddo a generalização do modelos. O WikiSQL, por sua vez, tinha bases de validação diferentes das bases de treinamento, porém as consultas eram muito simples.

Por isso, foi proposto o Spider, como alternativa de dataset que contém queries SQL de vários níveis e bases de dados de domínios distintos. O Spider apresenta 200 bases de dados com várias tabelas, 10,181 perguntas, com correspondentes 5,693 queries SQL. Também não tem sobreposição entre as bases de dados vistas no treinamento com as que usamos na validação. Assim, pelo Spider podemos treinar e validar o desempenho de um modelo diante de queries SQL complexas e mensurar a capacidade de generalização diante de domínios e requisições não vistos antes. Na tabela 1, temos a comparação com outros datasets

As perguntas da base de dados Spider são em Inglês. Portanto, antes de apresentá-las à entrada do modelo, para termos um tradutor de linguagem natural para SQL em Português, precisamos traduzi-las para Português.

4.2 Parsers

A interpretação das relações entre termos da frase e tipo delas é fundamental para que as traduzamos para as respectivas representações em SQL e, assim, tenhamos na saída a consulta que corresponda à intenção da frase. Tornar explícitas as interações entre os termos da frase, tanto a quem um termo se relaciona quanto à forma como é feito esse relacionamento (oposição, adição, entre outros) mostra-se uma ferramenta útil para tratar o problema da rede neural que usamos. Para gerar a informação sobre a sintaxe da frase, recorremos ao parser de dependências Spacy. A decomposição semântica, por sua vez, é feita pela biblioteca amrlib.

4.2.1 Parser de dependências

O parser de dependências é feito pela biblioteca em Python Spacy. Essa ferramenta permite realizar o parsing de dependências de frase em diversas línguas, com alta precisão e rapidamente. Além disso, também fornece a morfologia da frase e lematização de palavras.

Para todas essas tarefas, o Spacy tem pipelines pré-treinados para 25 línguas, incluindo Inglês e Português. A operação desses pipelines para realizar o processamento do texto requer apenas o download do próprio pipeline, cujas versões variam em tamanho e precisão nas tarefas, e importação da biblioteca em si, com o nome do respectivo pipeline como parâmetro na chamada da função de download. O pipeline de Inglês, na menor versão (cuja precisão nesse aspecto não varia muito para as maiores versões,o que justifica o seu uso), apresenta uma precisão na determinação das relações sintaticas da frase de 0.9, e o de Português, também na menor versão (também sem variar muito a precisão), é 0.8.

A saída do Spacy é uma árvore em formato tabular. A cada termo associamos uma linha, que tem não apenas a função sintática do termo, mas também os nós filhos dele e o nó pai (também podemos incluir a classe morfológica). Podemos ainda separar a frase não apenas termo a termo, mas por 'chunks' (fatias da frase que formam uma função sintática). A árvore pode ser visualizada por um display da biblioteca ou escrita no terminal. Na figura 4 temos uma árvore do Spacy para 'Autonomous cars shift insurance liability toward manufacturers'. A tabela abaixo mostra a saída do Spacy para a frase "Tenho em mim todos os sonhos do mundo".

```
Tenho ROOT Tenho VERB [mim, sonhos]
em case mim PRON []
mim obl Tenho VERB [em]
todos det sonhos NOUN [os]
os fixed todos DET []
sonhos obj Tenho VERB [todos, mundo]
do case mundo NOUN []
mundo nmod sonhos NOUN [do]
```

Nessa estrutura, o primeiro item de cada linha é o termo a que a linha corresponde. O segundo termo é a função sintática do termo, seguida pelo seu pai sintático (o termo do qual se origina a relação sintática do termo) e a classe morfológica do pai. Os termos no colchete são os filhos sintáticos do termo a que a linha se refere.

4.2.2 Parser semântico

O AMR por sua vez traz também as relações da frase (semânticas) por uma árvore, porém na notação PENMAN. Apenas os termos mais representativos da frase aparecem na representação. As relações entre os termos são feitas pelos argumentos do termo na árvore, que se desdobram recursivamente, com as dependências como argumentos do termo (e variáveis para nomear as ocorrências, por exemplo, "p"para nomear a primeira ocorrência de "pencil").

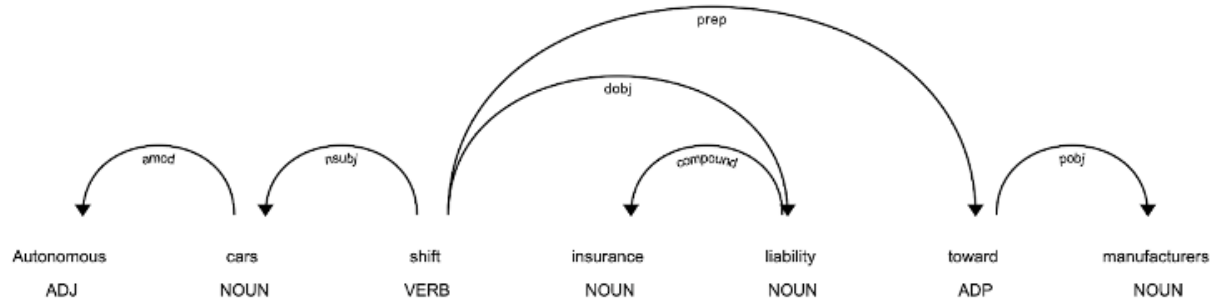

Figura 4 – O parsing da frase 'Autonomous cars shift insurance liability toward manufacturers' pelo Spacy

O parser de Abstract Meaning Representation (AMR) das frases é feito pela biblioteca Python amrlib. Com o parser, podemos gerar tanto o grafo AMR a partir de uma frase (sentence to graph), quanto uma frase a partir de um grafo AMR. Para isso, o parser é constituido por uma rede neural transformer, com vários modelos pré-treinados à disposição por tarefa, para que o usuário use o mais apropriado aos requisitos de tamanho, velocidade e precisão. A precisão dos modelos é medida em SMATCH (CAI; KNIGHT, 2013), uma medida da sobreposição entre duas estruturas de atributos semânticos, para avaliar a estrutura semântica da frase gerada: cada par AMR (referência e gerado) é traduzido para uma conjunção de proposições lógicas, que são comparadas entre si por meio de acurácia, revocação e F-score, sendo o maior F-score a pontuação SMATCH. O modelo que usamos no parser é um baseado no Bart-large, cuja frequência de inferência é de 17 por segundo, tamanho é 1.4 GB e tem SMATCH de 83.7. A árvore AMR produzida pelo parser amrlib é vista na seção 2.3.2.

5 Desenvolvimento do Trabalho

5.1 Tecnologias Utilizadas

A base de treinamento, com os pares de perguntas e respectivas consultas SQL, é a base Spider (YU et al., 2019), que também serve de *benchmark* para a tarefa de tradução de linguagem natural para SQL e foi vista na seção 4.1. A partir dessas perguntas dessa base de dados, uma vez traduzidas para Português, produzimos as informações relativas a essas frases, tanto sintáticas quanto semânticas, para que as concantenemos às respectivas frases originais e, assim, apresentemo-las como entrada para o nosso modelo, o RESDSQL.

A geração das informações sobre as frases de entrada será feita por dois parsers independentes, um para a informação sobre as dependências sintáticas na frase e outro para as relações semânticas. O parser para a sintaxe foi o Spacy e o amrlib para a representação semântica, como descrito na 4. Porém, antes de concatenar essas representações, que tem um formato tabular e de árvore, respectivamente com a entrada, é preciso linearizá-las, como descrevermos a seguir.

O Spacy, além de ser prático para uso, por requerer apenas o download do pacote para a língua alvo seguido pela importação da biblioteca em si, abrange várias línguas com boa precisão, incluisve Português. Além disso, ao contrário dos parsers de dependências tradicionais, gera na árvore as representações sintáticas propriamente, não apenas referentes à morfologia, o que atende aos nossos requisitos sobre o tipo de informação a apresentar.

O amrlib é uma biblioteca que gera a árvore AMR da frase de entrada por meio de uma rede neural Bart (LEWIS et al., 2019). O parsing é feito apenas pelo armazenamento do modelo neural na memória, a importação da biblioteca e uma chamada do modelo para realizar a inferência e gerar o AMR da frase. Porém, por causa das bases de dados para treinamento do modelo serem na maioria em inglês, o modelo faz o parsing apenas para essa língua de entrada. Assim, teríamos o parsing apenas em inglês.

As saídas dos dois parsers, porém, não podem ser apresentadas diretamente à rede, por terem estrturas diferentes daquela da frase original. Tanto a primeira, em formato tabular, quanto a segunda, em formato de árvore seguindo a notação PENMAN, precisam ser linearizadas, cada uma seguindo um método próprio, conforme descrevemos nas seções 5.1.2 e 5.1.1, a seguir.

No ranking do Spider, um dos modelos abertos no topo da classificação é o RESDSQL, acima do RAT-SQL, por isso esse será o modelo que refinaremos. O RESDSQL tem algumas opções de modelos de linguagem do tipo encoder-decoder, baseados na arquitetura T5, para usar na geração das queries, como T5 base ou large (RAFFEL et

al., 2023). Na implementação, testaremos o T5-base e um modelo de linguagem diferente, o Bart. O Bart será testado para que tenhamos uma arquitetura diferente do T5 nos resultados e, assim, tornem-se mais gerais.

5.1.1 Parser AMR

A maioria das bases de treinamento dos modelos de parsers para AMR são em Inglês, o que restringe os modelos que são treinados por elas a essa língua, tanto na frase quanto no grafo. É viável traduzir a frase de origem, porém para as representações no grafo, por usarem quadros de PropBank (PALMER; GILDEA; KINGSBURY, 2005; KINGSBURY; PALMER, 2002), que associa individualmente a função semântica que cada termo no seu vocabulário pode assumir a um grupo de papéis semânticos, representados por números, requeririam a adaptação dessa notação para a língua alvo, o que não é direto, uma vez que um verbo em Inglês pode assumir mais sentidos do que usa tradução direta em Português (mapear para mais de um termo em Português). Para que possamos ainda assim ter a representação semântica para as frases, faremos o parsing da frase em Inglês e concatenar a respectiva representação AMR, proveniente do parser e linearizada, com a frase traduzida em Português.

A árvore AMR tem um formato no qual os nós são separados por parênteses, o que nos facilita por servir de separador natural para a nossa rede neural, assim apenas removemos os espaços embutidos entre os termos da árvore de saída para termos a sua linearização. Para cada pergunta da base de dados, faremos esse processo fabricando uma base de dados que inclui essa informação. Abaixo, um exemplo de pergunta nessa nova base de dados (incluindo o AMR da frase):

```
"Liste todos os nomes de músicas por cantores acima da idade média.
(l / list - 01 :ARG1 (n / name - 01 :ARG0 (p / person :ARG0 - of
(s / sing - 01) :mod (a / above :op1 (a2 / age - 01 :ARG1 - of
(a3 / average - 04)))) :ARG1 (s2 / song) :mod (a4 / all)) "
```

5.1.2 Parser de dependências

Para cada frase do banco de dados, fazemos o parsing sintático dela usando o Spacy, concatenando à frase original em Português uma representação da sua sintaxe. Essa representação é feita pela inclusão apenas dos termos sintaticamente relevantes da frase, como sujeito, objeto direto e indireto e alguns termos relacionados, e contém apenas o texto do termo, a sua função sintática na frase. Para que tenhamos o mesmo padrão de resgistro desses termos, que são proibitivamente escritos em Inglês na representação AMR por causa dos quadros de PropBank, também teremos os termos sintáticos em Inglês.

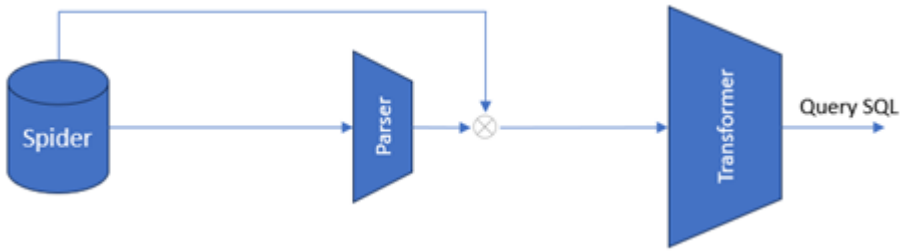

Figura 5 – Processo de treinamento dos modelos de linguagem, T5, ou Bart, com as informações sintáticas ou semânticas

As informações referentes aos termos são separadas por um separador "[row]". Assim, temos uma base que apresenta a informação sintática da frase. Abaixo, temos um exemplo de uma frase no banco de dados:

```
"Liste todos os nomes de músicas por cantores acima da idade média.
[row] names; dobj [row] singers; pobj [row] age; pobj"
```

O refinamento do modelo sintático seria investigar quais relações são relevantes ou não para apresentar ao modelo. Restringimos apenas aos termos que certamente são importantes, como sujeito e objetos, e termos relacionados diretamente a eles.

5.2 Projeto e Implementação

O treinamento dos modelos foi feito usando dois modelos de linguagem diferentes, T5 e Bart, com o intuito de investigar o impacto das informações para diferentes arquiteturas de transformers. A cada modelo de linguagem usado, Bart, ou T5, treinamos um representante de referência usando apenas a base de dados Spider traduzida para Português, como benchmark para os treinamentos com as informações. Além desse, treinamos modelos usando ou a base com as informações sintáticas ou semânticas, a depender do tipo que queremos avaliar. Na figura 5, vemos a representação do processo de treinamento dos modelos. Na figura, o parser pode ser tanto o sintático quanto o semântico, a depender do tipo de informação que queremos apresentar no treinamento. O operador após o parser significa concatenação. Os treinamentos foram feitos por 32 e 128 épocas, independentemente, ou seja foram dois processos de treinamento para cada modelo, a fim de verificar a evolução do aprendizado dos modelos.

Modelo	Exact-set-Match accuracy (EM)	EXecution accuracy (EX)
T5 sem informação	0.55029	0.57543
T5 com informação sintatica	0.59090	0.62475
T5 com AMR	0.63926	0.68181
Bart sem informação	0.18181	0.18085
Bart com AMR	0.22147	0.24468
Bart com informação sintatica	0.21566	0.22050

Tabela 2 – Avaliação dos modelos após treinamento por 32 épocas

Modelo	Variação relativa - EM	Variação relativa - EX
T5 com informação sintatica	0.07381	0.07563
T5 com AMR	0.1616	0.18487
Bart com AMR	0.2182	0.3529
Bart com informação sintatica	0.1862	0.2192

Tabela 3 – Variações relativas das pontuações dos modelos ao benchmark (sem informação) para 32 épocas

5.3 Testes e Avaliação

A avaliação do modelo é feita usando duas métricas, conforme (YU et al., 2019) e (ZHONG; YU; KLEIN, 2020). A primeira métrica é Exact-set-Match accuracy (EM), na qual comparamos a query gerada pelo modelo com a 'query ouro', ou seja a query referência pela base de dados. Para isso, convertemos a consulta para uma estrutura especial de dados e verificamos se há uma correspondência exata entre as duas. A segunda métrica é EXecution accuracy (EX), na qual comparamos os resultados de execução da 'query' gerada pelo modelo com o resultado da 'query ouro'. A segunda métrica é sensível a ruído, nominalmente a falsos positivos, por comparar apenas valores e não a origem dos valores. A primeira, em contrapartida, é pouco sensível, por ter mais falsos negativos, uma vez que uma consulta pode ser correta para a questão mesmo que não tenha a mesma estrutura da 'query ouro'. Assim, para avaliar o modelo usamos a soma das duas métricas, para todos checkpoints. Nas tabelas 2 e 4, temos os melhores resultados para 32 épocas e 128 épocas de treinamento para cada métrica, respectivamente. Os treinamentos para 32 épocas e 128 épocas foram feitos separadamente, por isso é possível que alguns resultados sejam maiores para 32 épocas do que para 128 épocas.

Notamos ainda pelas tabelas de variações relativas, principalmente 3, que, para alguns modelos, como o Bart com AMR treinado para 32 épocas, a variação relativa seria ainda mais expressiva se comparada com o modelo gerado no treinamento completo de 128 épocas quando este estava na época 32, com ganhos de 0.40 e 0.52. Porém, como avaliamos nas tabelas de 32 épocas modelos que assim foram treinados (não os que foram por 128 épocas), não mostramos nelas essas variações, mas apenas aqui.

Modelo	Exact-set-Match accuracy (EM)	EXecution accuracy (EX)
T5 sem informação	0.586073500967118	0.60155
T5 com informação sintatica	0.62959	0.64894 (0.65184)
T5 com AMR	0.65280 (0.65377)	0.68665 (0.69342)
Bart sem informação	0.16344 (0.16537)	0.17118
Bart com informação sintatica	0.21760	0.22243
Bart com AMR	0.18859	0.19632 (0.19729)

Tabela 4 – Avaliação dos modelos após treinamento de 128 épocas

Modelo	Variação relativa - EM	Variação relativa - EX
T5 com informação sintatica	0.07425	0.0787 (0.0836)
T5 com AMR	0.1138 (0.1155)	0.1414 (0.1527)
Bart sem informação	0.16344 (0.16537)	0.17118
Bart com informação sintática	0.3313 (0.3158)	0.29946
Bart com AMR	0.1538 (0.1403)	(0.1526)

Tabela 5 – Variações relativas das pontuações para as duas métricas dos modelos ao benchmark (sem informação) para 128 épocas

As pontuações dos modelos vistas nas tabelas 2 e 4 mostram que a infusão de conhecimento sintático ou semântico no treinamento para a tarefa é relevante para o aprendizado. Os modelos, independentemente da arquitetura (T5 e Bart), que foram apresentados à sintaxe ou representação semântica da frase tiveram, tanto para um número de épocas baixo (32), quanto para um número de épocas alto (128), desempenho acima do benchmark, isto é, acima do seu par treinado sem informação alguma. Isso foi visto para todas arquiteturas na tabela. Além disso, para as duas tabelas, a informação semântica foi a que se mostrou mais relevante para o modelo, apresentando os maiores ganhos relativos ao benchmark, para ambas arquiteturas.

É importante ressaltar que, além de os modelos que foram apresentados no treinamento às informacões sintáticas ou semânticas terem as maiores pontuações nas tabelas, esses modelos, ainda que treinados por apenas 32 épocas, apresentam resultados melhores do que o benchmark treinado por 128 épocas. Ou seja, a apresentação do conhecimento sintático ou semântico ao modelo torna o processo de aprendizado não apenas melhor, permitindo que tenha uma pontuação maior do que teria se não a fizéssemos, mas também mais eficaz, gerando resultados relevantes (até mesmo maiores) para um treinamento por menos épocas do que as pontuações registradas pelos modelos que foram treinados sem informação por 4 vezes mais épocas.

6 Considerações Finais

6.1 Conclusões do Projeto de Formatura

A infusão de conhecimento sintático ou semântico , particularmente para uma tarefa que envolve uma linguagem estruturada, mostrou-se relevante para tornar mais eficiente o processo de aprendizado dos transformers pelo trabalho. No modelo apresentado no trabalho, realizamos a infusão de conhecimento sintático (ou semântico) sobre a frase de entrada para o modelo para aumentar a interpretação da rede neural sobre a real intenção da frase, traduzida pelos termos que se relacionam na frase e pela natureza dessa interação. Assim, na decodificação para a saída, geramos resultados que respeitam tais relações e mais conscientes delas, transmitidas pela entrada.

Pelas tabelas 2 e 4, o processo mostrou-se relevante para o transformer, com os modelos que foram apresentados às informações, tanto sintáticas, quanto semânticas, revelando pontuações maiores do que seus pares com treinamento convencional, para dois modelos de linguagem diferentes. Particularmente, notamos que a semântica da frase é mais importante para os transformers, uma vez que os resultados mais significativos foram dos modelos que usaram esse tipo de representação (ou informação).

Ainda, os modelos que tiveram essa apresentação no treinamento à sintaxe ou semântica da frase alcançaram, em 32 épocas, resultados melhores do que seus pares sem essa infusão após treinamento por 128 épocas e até mesmo próximos das suas próprias pontuações após treinamento por 128 épocas também. Ou seja, o processo não apenas permitiu que os modelos alcançassem resultados que não seriam vistos sem essa estratégia, mas também de forma mais rápida, economizando tempo e recursos computacionais, o que, em um período no qual a demanda por recursos energéticos para satisfação de treinamentos massivos se tornou tamanha a ponto de não podermos ignorar o impacto ambiental produzido pelo treinamento desses modelos (pegadas ecológicas), torna-se uma alternativa importante e necessária quando desenvolvemos novos modelos, alinhando-a com a computação verde.

6.2 Contribuições

Além da revelação de que a infusão sintática ou semântica nos modelos de linguagem são relevantes para tornar o aprendizado melhor e mais eficiente, para linguagens estruturadas, tivemos como produto do trabalho modelos de tradução de linguagem natural para SQL que marcam novo estado da arte (apenas na versão base do T5, com 0.6528 na

primeira métrica) e que geram consultas em SQL com os respectivos valores mencionados na pergunta, algo que não tínhamos até então.

6.3 Perspectivas de Continuidade

Como continuação do projeto podemos investigar para novas tarefas, igualmente sensíveis, sujeitas a grandes variações nos resultados por pequenas alterações na frase, o impacto dessas informações, sintática e semântica, no treinamento. Tais tarefas podem ser tradução de uma língua para outra, na qual precisamos ter mesmas associações feitas na língua fonte feitas na língua alvo, produção de linhas de código a partir de uma descrição, na qual o código resultante precisa respeitar as restrições impostas pela frase de entrada (que são vistas pelas relações entre termos da frase e tipo dessas relações), ou até mesmo geração de imagens a partir de uma frase, por ser imprescindível a correta associação entre os termos da frase para produzir a imagem pretendida. A estratégia mostra-se importante e promissora para tornar mais eficiente tais tarefas.

Referências

AFFOLTER, K.; STOCKINGER, K.; BERNSTEIN, A. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, Springer Science and Business Media LLC, v. 28, n. 5, p. 793–819, ago. 2019. ISSN 0949-877X. Disponível em: <http://dx.doi.org/10.1007/s00778-019-00567-8>. Citado na página 22.

BAI, J. et al. *Syntax-BERT: Improving Pre-trained Transformers with Syntax Trees*. 2021. Citado na página 22.

BAI, J. et al. Syntax-bert: Improving pre-trained transformers with syntax trees. 3 2021. Disponível em: <http://arxiv.org/abs/2103.04350>. Citado na página 29.

BANARESCU, L. et al. Abstract Meaning Representation for sembanking. In: PAREJA-LORA, A.; LIAKATA, M.; DIPPER, S. (Ed.). *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Sofia, Bulgaria: Association for Computational Linguistics, 2013. p. 178–186. Disponível em: <https://aclanthology.org/W13-2322>. Citado na página 31.

CAI, S.; KNIGHT, K. Smatch: an evaluation metric for semantic feature structures. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. [S.l.: s.n.], 2013. p. 748–752. Citado na página 40.

CURREY, A.; HEAFIELD, K. *Incorporating Source Syntax into Transformer-Based Neural Machine Translation*. 2019. 24-33 p. Citado na página 30.

DEVLIN, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. Citado na página 28.

GAO, D. et al. *Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation*. 2023. Citado na página 20.

IACOB, R. C. A. et al. Neural approaches for natural language interfaces to databases: A survey. In: SCOTT, D.; BEL, N.; ZONG, C. (Ed.). *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020. p. 381–395. Disponível em: <https://aclanthology.org/2020.coling-main.34>. Citado na página 19.

JOSÉ, M. A.; COZMAN, F. G. mrat-sql+gap: A portuguese text-to-sql transformer. In: BRITTO, A.; DELGADO, K. V. (Ed.). *Intelligent Systems*. Cham: Springer International Publishing, 2021. p. 511–525. ISBN 978-3-030-91699-2. Citado na página 20.

KIM, H. et al. Natural language to sql: Where are we today? *Proc. VLDB Endow.*, VLDB Endowment, v. 13, n. 10, p. 1737–1750, jun 2020. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/3401960.3401970>. Citado na página 22.

KINGSBURY, P. R.; PALMER, M. From treebank to propbank. In: *LREC*. [S.l.: s.n.], 2002. p. 1989–1993. Citado 2 vezes nas páginas 31 e 42.

LEIVADA, E.; MURPHY, E.; MARCUS, G. *DALL-E 2 Fails to Reliably Capture Common Syntactic Processes*. 2022. Citado na página 19.

LEWIS, M. et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. Citado na página 41.

LI, F.; JAGADISH, H. V. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, VLDB Endowment, v. 8, n. 1, p. 73–84, sep 2014. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/2735461.2735468>. Citado na página 25.

LI, H. et al. *RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL*. 2023. Citado na página 20.

LIU, Y. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. Citado na página 28.

OZCAN, F. et al. State of the art and open challenges in natural language interfaces to data. In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2020. (SIGMOD '20), p. 2629–2636. ISBN 9781450367356. Disponível em: <https://doi.org/10.1145/3318464.3383128>. Citado na página 22.

PALMER, M.; GILDEA, D.; KINGSBURY, P. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, v. 31, n. 1, p. 71–106, 03 2005. ISSN 0891-2017. Disponível em: <https://doi.org/10.1162/0891201053630264>. Citado 2 vezes nas páginas 31 e 42.

PRICE, P. Evaluation of spoken language systems: The atis domain. In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*. [S.l.: s.n.], 1990. Citado na página 37.

PUNYAKANOK, V.; ROTH, D.; YIH, W.-T. *The Importance of Syntactic Parsing and Inference in Semantic Role Labeling*. 2008. Disponível em: <http://direct.mit.edu/coli/article-pdf/34/2/257/1798602/coli.2008.34.2.257.pdf>. Citado na página 29.

RAFFEL, C. et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. Citado na página 42.

SAHA, D. et al. Athena: An ontology-driven system for natural language querying over relational data stores. *Proc. VLDB Endow.*, VLDB Endowment, v. 9, n. 12, p. 1209–1220, aug 2016. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/2994509.2994536>. Citado na página 25.

SCHOLAK, T.; SCHUCHER, N.; BAHDANAU, D. *PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models*. 2021. Citado na página 26.

WANG, B. et al. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In: JURAFSKY, D. et al. (Ed.). *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020. p. 7567–7578. Disponível em: <https://aclanthology.org/2020.acl-main.677>. Citado na página 19.

WILCOX, E. et al. Structural supervision improves learning of non-local grammatical dependencies. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 3302–3312. Disponível em: <https://aclanthology.org/N19-1334>. Citado na página 29.

WU, Z.; PENG, H.; SMITH, N. A. Infusing finetuning with semantic dependencies. *Transactions of the Association for Computational Linguistics*, v. 9, p. 226–242, 3 2021. ISSN 2307-387X. Disponível em: <https://doi.org/10.1162/tacl_a_00363>. Citado na página 30.

YU, T. et al. *TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation*. 2018. Citado na página 26.

YU, T. et al. *Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task*. 2019. Citado 3 vezes nas páginas 19, 41 e 44.

ZANZOTTO, F. M. et al. KERMIT: Complementing transformer architectures with encoders of explicit syntactic interpretations. In: WEBBER, B. et al. (Ed.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020. p. 256–267. Disponível em: <https://aclanthology.org/2020.emnlp-main.18>. Citado na página 22.

ZELLE, J. M.; MOONEY, R. J. Learning to parse database queries using inductive logic programming. In: *Proceedings of the national conference on artificial intelligence*. [S.l.: s.n.], 1996. p. 1050–1055. Citado na página 37.

ZHANG, Z. et al. Semantics-aware bert for language understanding. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2020. v. 34, n. 05, p. 9628–9635. Citado na página 30.

ZHANG, Z. et al. *SG-Net: Syntax-Guided Machine Reading Comprehension*. 2020. Disponível em: <www.aaai.org>. Citado na página 30.

ZHONG, R.; YU, T.; KLEIN, D. Semantic evaluation for text-to-SQL with distilled test suites. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020. p. 396–411. Disponível em: <https://aclanthology.org/2020.emnlp-main.29>. Citado na página 44.

ZHONG, V.; XIONG, C.; SOCHER, R. *Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning*. 2017. Citado na página 37.