Avaliação da aplicabilidade do modelo SQLNet na conversão de Consultas em Linguagem Natural para Consultas SQL a uma Base Integrada Corporativa

Henrique Bueno Rodrigues
Instituto de Computação – Universidade Federal Fluminense (UFF)
henriquebueno@id.uff.br

Abstract

O problema conhecido como Semantic Parsing consiste no processo de tradução de um conteúdo expresso em linguagem natural para uma representação formal que uma máquina consiga processar. Uma especialização desse problema, chamada de Natural Language to SQL (NL2SQL), é a tarefa de converter uma consulta em linguagem natural para SQL. Esse trabalho avaliou a aplicação de um modelo NL2SQL chamado SQLNet na conversão de consultas a uma Base Integrada Corporativa.

1 Introdução

Apesar do surgimento de diversas tecnologias de bancos de dados não convencionais, por exemplo, NoSQL [8] e Hadoop [9], sistemas de corporações pequenas, médias e grandes ainda possuem enorme acoplamento e dependência de bancos de dados relacionais, tais como Oracle [10] e SQL Server [11].

Em paralelo a isso, surge uma grande demanda de Democratização da Informação [12] por parte de usuários corporativos que não conhecem a linguagem SQL e que desejam explorar seus dados além das limitações das interfaces de sistemas.

Nesse contexto, o problema conhecido como Natural Language to SQL (NL2SQL) [2], que é um subtipo de problema SEQ2SEQ [3][4], consiste na tarefa de converter uma consulta em linguagem natural para SQL, tem sua relevância ampliada e recebe maior atenção de pesquisadores. Um exemplo disso é o ranking WikiSQL [1] que tem recebido submissões de trabalhos com alta frequência.

Um dos artigos que evoluiu os resultados sobre o dataset WikiSQL é o SQLNet [2], que propôs a geração de consultas SQL a partir de consultas em linguagem natural sem o uso de aprendizado por reforço.

A proposta desse trabalho é avaliar a aplicação da solução apresentada em SQLNet na geração de consultas SQL a partir de consultas em linguagem natural sobre uma Base de Dados Integrada Corporativa (BDIC). Seguem algumas características dessa base:

- Possui 59 tabelas;
- A tabela com maior quantidade de colunas tem 53 colunas;
- A tabela com menor quantidade de colunas tem 2 colunas;
- A média de colunas por tabela é 12.

As próximas seções detalharão as atividades que foram desempenhadas para alcançar o objetivo apresentado nos parágrafos anteriores. A seção 2 apresenta uma avaliação do WikiSQL. A seção 3 apresenta uma avaliação do SQLNet. A seção 4 apresenta as atividades desempenhadas para a criação de um dataset de consultas para BDIC. A seção 5 avaliou uma rede neural de tradução de textos em português para inglês. A seção 6 apresenta a conclusão e ideias de trabalhos futuros.

2 WikiSQL

O trabalho [1] apresenta SEQ2SQL, uma rede neural profunda que tem o objetivo de traduzir perguntas em linguagem natural para consultas SQL usando aprendizado por reforço.

Uma outra contribuição desse trabalho é o dataset WikiSQL que tem 80654 registros com pares de perguntas em linguagem natural e consultas em SQL. Esse dataset foi anotado manualmente através de crowdsourcing utilizando a infraestrutura Amazon Mechanical Turk [6].

Diversos pesquisadores se interessaram em avaliar o desempenho de seus modelos no WikiSQL. A Tabela 1 [13] apresenta uma lista de modelos e seus resultados sobre o dataset em questão.

Foram considerados dois pontos na avaliação dos modelos: (a) forma lógica do SQL gerado (colunas 2 e 4) e (b) resultado da execução da query SQL gerada (colunas 3 e 5).

O modelo “Wang 2017” utilizou o conteúdo das tabelas para o treinamento do modelo. Esse é um ponto de atenção, uma vez que pode violar restrições de sigilosidade das informações armazenadas nas bases de dados. Outro problema no uso do conteúdo das tabelas é a escalabilidade, uma vez que

tabelas muito grandes podem aumentar consideravelmente o tempo da fase de treinamento.

Modelo	Acurácia forma lógica dsv	Acurácia execução dsv	Acurácia forma lógica tst	Acurácia execução tst
Coarse2Fine (Dong 2018)	72.5	79.0	71.7	78.5
TypeSQL (Yu 2018)	-	74.5	-	73.5
PT-MAML (Huang 2018)	63.1	68.3	62.8	68.0
SQLNet (Xu 2017)	-	69.8	-	68.0
Wang 2017	62.0	67.1	61.5	66.8
Seq2SQL (Zhong 2017)	49.5	60.8	48.3	59.4
Baseline (Zhong 2017)	23.3	37.0	23.4	35.9

Tabela 1. Lista de modelos e seus resultados para o dataset WikiSQL

O modelo SEQ2SQL, que foi apresentado em conjunto com o dataset WikiSQL, gerou os primeiros resultados do ranking em agosto de 2017. Em pouco tempo diversos trabalhos evoluíram os resultados e atualmente o modelo Coarse2Fine, que foi proposto em maio de 2018, apresenta a melhor acurácia de execução para a base de teste (71.7%).

Um ponto de destaque do WikiSQL é que as tabelas citadas no dataset são aleatoriamente distribuídas nas bases de treinamento, desenvolvimento e teste, de tal forma que cada tabela só aparece em uma única base (treinamento, desenvolvimento ou teste). Essa característica representa um desafio considerável para os modelos, pois demanda que os modelos sejam capazes de generalizar a capacidade de gerar SQLs, independente dos schemas dos modelos.

2.1 Avaliação do WikiSQL

Para avaliar o WikiSQL foram realizadas as seguintes etapas:

- Estudo da estrutura do WikiSQL: o dataset foi disponibilizado nos formatos jsonl e db (SQLite3). A Figura 1 apresenta um exemplo de registro do dataset WikiSQL. O campo phase representa uma informação interna do processo de geração do conjunto de dados, que foi dividido em fases. O campo question representa a pergunta em linguagem natural. O campo sql representa a consulta em sql. O campo sql é dividido nos campos conds, sel e agg. O campo conds tem 3 valores, onde o primeiro indica o índice da coluna que aparece na restrição, o segundo indica o índice do operador e o terceiro indica o valor a ser comparado com o operando. O campo sel indica o índice da coluna que será selecionado. O campo agg representa o operador (índice do operador na lista de operadores) de agregação, por exemplo, sum e avg. Por fim, o campo table_id identifica a tabela que será consultada.

- Execução de testes sobre o modelo SEQ2SQL: A rede neural SEQ2SQL é um modelo inicial para a avaliação do dataset. Esse modelo foi implementado usando Python 3 e a biblioteca PyTorch [14]. O modelo gerado pelo treinamento descrito no artigo foi disponibilizado pelos autores. Dessa forma, foi possível instalar o modelo e executar alguns testes.


```
{
  "phase":1,
  "question":"who is the manufacturer for the order year 1998?",
  "sql":{
    "conds":[
      [
        0,
        0,
        "1998"
      ]
    ],
    "sel":1,
    "agg":0
  },
  "table_id":"1-10007452-3"
}
```


Figura 1. Exemplo de um registro do WikiSQL

3 SQLNet

O trabalho [2] propôs um modelo para o dataset WikiSQL. Diferentemente de outros modelos de parsing semântico que são projetados para serem agnósticos em relação à gramática de saída, a ideia básica do SQLNet é utilizar um sketch para a gramática do SQL. Dessa forma, o principal trabalho é preencher as lacunas do sketch ao invés de ter que prever tanto a gramática de saída quanto o conteúdo. A figura 2 ilustra o sketch utilizado.

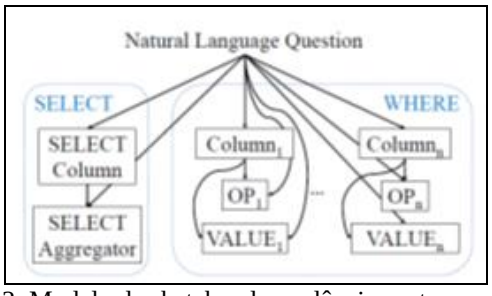

Figura 2. Modelo de sketch e dependências entre os componentes

Diferentemente do modelo SEQ2SQL, esse modelo não utilizou a técnica de aprendizado por reforço e obteve resultados superiores (próximos a 70%).

3.1 Avaliação do SQLNet

O código da rede SQLNet foi disponibilizado pelos autores no GitHub [7]. Essa rede foi desenvolvida em Python 2.7 utilizando a biblioteca PyTorch. Seguem alguns pontos observados durante o processo de execução do código disponibilizado:

- Analisando o código, observei que ele estava preparado para rodar com CPU e GPU. Entretanto, ao rodar o treinamento com CPU, ocorreram alguns erros que aparentemente eram simples e que, teoricamente, evidenciaram que o código não foi testado com CPU.
- O código de verificação da disponibilidade de GPU no hardware alvo da execução não estava funcionando. Assim, ele não identificava GPUs e mesmo assim passava pelo trecho de código previsto para hardware com GPU, gerando erro na execução.
- O software foi desenvolvido em Python 2.7. Isso gerou alguns problemas com dependências de outras versões de bibliotecas que utilizam Python 3.
- Os autores disponibilizaram o código, porém não liberaram o modelo treinado. Entrei em contato com os autores solicitando o modelo treinado (*issue* criada no *GitHub*), mas não obtive resposta.

O artigo que apresenta o SQLNet diz que o modelo foi treinado em 200 épocas. Dessa forma, como o modelo não foi disponibilizado pelos autores, executei o processo de treinamento para esse período.

O principal objetivo foi fazer uma prova de conceito do SQLNet e não tentar aprimorar os resultados obtidos. Para isso, o primeiro passo foi rodar o treinamento da rede SQLNet por 200 épocas, conforme especificado no artigo.

O processo de treinamento foi executado em CPU, o que aumentou consideravelmente a duração desta etapa. O processo durou 100 épocas e foi interrompido pelo próprio algoritmo de treinamento. O tempo total de horas foi xxx.

O segundo passo foi gerar o arquivo que representa a base de dados que se deseja testar. Dessa forma, foi preciso criar um arquivo com as informações da estrutura da tabela que foi utilizada no teste conforme a especificação do formato detalhado pelo SQLNet.

O terceiro passo foi gerar o arquivo que representa as consultas que se deseja testar no modelo.

Por fim, o último passo foi executar o modelo com as consultas geradas no passo 3. Durante essa etapa houve um erro no processamento do arquivo com as perguntas em linguagem natural. Esse erro ainda está sendo investigado.

4 Proposta de criação de um dataset

Conforme o relatório Data Scientist Report 2017 [5], para 53% dos entrevistados, a atividade que mais consome tempo dos cientistas de dados é coletar, rotular, limpar e organizar dados. Para que técnicas de aprendizado de máquina sejam aplicadas com eficácia, é primordial que os dados que serão utilizados como entrada sejam trabalhados e tenham uma boa representatividade do modelo que se deseja prever.

Para os problemas da classe NL2SQL essa questão não é diferente. Dessa forma, para que um modelo de aprendizado de máquina consiga converter consultas em linguagem natural escritas em português para consultas em SQL, é fundamental que exista um conjunto de dados expressivo que ilustre diferentes cenários.

Uma das propostas desse trabalho é a criação de um dataset com pares <consultas em linguagem natural, consultas SQL> sobre uma base de dados chamada Base de Dados Integrada Corporativa (BDIC). A tabela 1 ilustra parte da estrutura dessa base.

Cada linha da Tabela 1 é uma combinação <tabela, coluna, tipo coluna, operador> que representa um operador (EQUAL, NOT_EQUAL, LESS, GREATER, LESS_OR_EQUAL, GREATER_OR_EQUAL, BETWEEN, NULL, NOT NULL, LIKE e NOT LIKE) suportado pelo tipo de uma coluna de uma determinada tabela. Através da Tabela 1 é possível gerar diversas combinações de consultas SQL sobre a BDIC.

Para a geração do dataset desse trabalho, foi proposto um formato padrão para as consultas baseado no formato de consultas do dataset WikiSQL. A Figura 3 ilustra o formato padrão proposto.

TABELA	COLUNA	TIPO DA COLUNA	OPERADOR
País	Código	Número	EQUAL
País	Código	Número	NOT_EQUAL
País	Código	Número	GREATER
País	Código	Número	LESS
...	...	...	...
País	Sigla	Texto	EQUAL
País	Sigla	Texto	NOT_EQUAL
País	Sigla	Texto	LIKE
...	...	...	...

Tabela 2: Representação das tabelas da BDIC

```
SELECT COLUNA (, COLUNA)*
FROM TABELA
WHERE COLUNA OP VALOR (AND COLUNA OP VALOR)*
```

Figura 3. Formato das consultas do dataset

Onde os tokens COLUNA, TABELA, OP e VALOR representam:

- TABELA: token que identifica uma tabela. Não depende de nenhum outro token.
- COLUNA: token que identifica uma coluna. Depende do valor atribuído ao token TABELA.
- OP: token que identifica um operador. Depende do valor atribuído ao token COLUNA.
- VALOR: token que identifica um valor. Depende do valor atribuído aos tokens COLUNA e OPERADOR.

Um ponto de atenção que também aparece no WikiSQL é a limitação das consultas acessarem uma única tabela. Os autores do WikiSQL comentaram que o objetivo é evoluir essa limitação em trabalhos futuros. Atualmente, uma forma de minimizar esse problema é criar estruturas de views [17], que podem ser utilizadas para combinar colunas de diferentes tabelas. Assim, as cláusulas join do SQL ficariam transparentes para o modelo de predição que só veria o resultado da consulta como se estivesse acessando uma única tabela.

A partir da descrição das tabelas da BDIC (Tabela 1) e do formato padrão das consultas SQL (Figura 3), é possível escrever um algoritmo para gerar um conjunto de consultas. O procedimento descrito na Figura 4 apresenta o pseudocódigo para a geração de um conjunto de consultas a serem executadas sobre as tabelas da BDIC.

```
Procedimento de criação de registros para dataset{
    Para cada tabela T da BDIC, faça{
        As colunas a serem utilizadas nas cláusulas "select" e "where" do
        comando SELECT que será criado serão independentes?
        Sim{
            1: Escolha a quantidade K de colunas da cláusula select;
            2: Escolha K colunas da tabela T; OBS: as quantidades de
            colunas das tabelas variam;
            3: Escolha Q a quantidade de restrições da cláusula
            "WHERE";
            4: Escolha Q colunas da tabela T;
            5: Para cada uma das Q colunas, escolha P operadores;
            OBS: os valores serão fixos.
        }
        Não{
            6: Igual ao item 1; OBS: nesse item é definido o valor de
            K.
            7: Igual ao item 2;
            8: Escolha Q a quantidade de restrições da cláusula
            "WHERE" onde 1<=Q<=K;
            9: Escolha Q colunas da tabela T onde esse conjunto seja
            subconjunto das colunas escolhidas em 7;
            10: Igual ao 5.
        }
    }
}
```

Figura 4. Procedimento para geração de consultas SQL

A proposta desse procedimento é realizar diversas combinações entre as colunas das tabelas do BDIC para a geração das consultas SQL.

Entretanto, para a conclusão do dataset ainda é preciso que, para cada uma das consultas SQL geradas, haja uma representação dela em linguagem natural. Apesar de ser possível gerar as consultas relacionadas em linguagem natural, essa abordagem torna o dataset extremamente pobre já que através dele não será possível criar modelos gerais que consiga compreender as diferentes formas de se escrever uma consulta em linguagem natural, mas ele ficará viciado na forma que as consultas em português foram geradas automaticamente.

Uma abordagem para a geração de consultas em linguagem natural para as consultas geradas em SQL é a estratégia colaborativa. Nesse formato, pessoas são remuneradas por cada colaboração. A Amazon disponibiliza uma infraestrutura para a publicação desses projetos de colaboração, chamada Amazon Mechanical Turk [6].

Em função das limitações de tempo, não foi possível evoluir a proposta de geração do dataset. Esse ponto será abordado na última seção desse trabalho que discute as conclusões e trabalhos futuros.

5 Avaliação de uma API de tradução

O trabalho [15] apresenta uma API para tradução de textos baseada no Google Tradutor [16]. O objetivo é utilizar essa API para processar as perguntas em linguagem natural escritas em português, traduzi-las para inglês e posteriormente repassá-las para o modelo SQLNet.

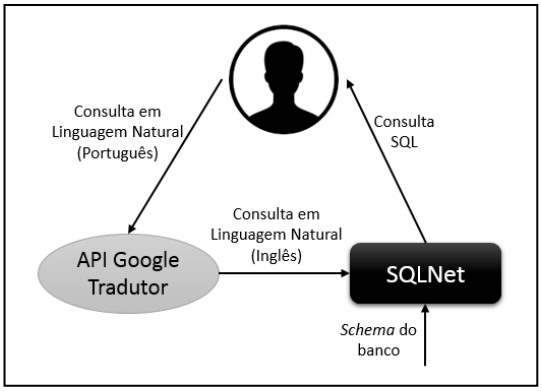

Figura 5. Arquitetura de Integração entre API Google Tradutor e SQLNet

A Figura 5 ilustra uma proposta de arquitetura de integração simplificada entre a API do Google Tradutor e o modelo SQLNet.

Através do uso dessa arquitetura será possível permitir que o usuário especifique suas consultas em português, ao invés de obrigá-lo a traduzi-las para inglês, como era a proposta inicial. Entretanto, em função de limitações de tempo, não foi possível avançar nessa avaliação.

6 Conclusão e Trabalhos Futuros

Este trabalho apresentou uma visão geral do *dataset* WikiSQL e da rede neural SQLNet que tem como objetivo converter consultas em linguagem natural para consultas SQL. Consegui rodar com minhas perguntas?

Além disso, também foi discutida uma estratégia para a geração de um *dataset* para uma base de dados específica, chamada BDIC. A geração desse *dataset* foi dividida em duas etapas: (1) geração das consultas SQL e (2) escrita das consultas em linguagem natural para cada um dos comandos *select* definidos na etapa 1. O código de geração dos comandos SQL foi desenvolvido em Java e disponibilizado no GitHub do projeto. Entretanto, não foi possível concluir a etapa 2 dentro do prazo estipulado.

Como próximos trabalhos, espera-se evoluir o *dataset* para o BDIC e avaliar outros modelos que tratam o problema NL2SQL.

O material produzido neste trabalho está em https://github.com/riquebueno/trabalhoIA2.

Agradecimentos

Agradeço à Professora Aline Paes, aos meus colegas de turma e à minha família: Cecília e Nathalia.

Referências

[1] Zhong, Victor, Caiming Xiong, and Richard Socher. "Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning." arXiv preprint arXiv:1709.00103 (2017).
[2] Xu, Xiaojun, Chang Liu, and Dawn Song. "SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning." arXiv preprint arXiv:1711.04436 (2017).
[3] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.
[4] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).
[5] Site https://visit.crowdflower.com/WC-2017-Data-Science-Report_LP.html acessado em 12/06/2018.
[6] Site https://www.mturk.com acessado em 12/06/2018.
[7] Site https://github.com/xiaojunxu/SQLNet acessado em 12/06/2018.
[8] Site https://pt.wikipedia.org/wiki/NoSQL acessado em 12/06/2018.
[9] Site http://hadoop.apache.org acessado em 12/06/2018.
[10] Site https://www.oracle.com/br/database/index.html acessado em 12/06/2018.
[11] Site https://www.microsoft.com/en-us/sql-server/sql-server-2017 acessado em 12/06/2018.
[12] Site https://www.forbes.com/sites/bernardmarr/2017/07/24/what -is-data-democratization-a-super-simple-explanation-and-the-key-pros-and-cons acessado em 12/06/2018.
[13] https://github.com/salesforce/WikiSQL acessado em 12/06/2018.
[14] Site https://pytorch.org acessado em 12/06/2018.
[15] Site https://github.com/matheuss/google-translate-api acessado em 19/06/2018.
[16] Site https://translate.google.com acessado em 19/06/2018.
[17] Site https://en.wikipedia.org/wiki/View_%28SQL%29 acessado em 15/06/2018.