Alberto Marengo

# Capstone Project: Final Report
# TEXT TO SQL: Using ML to generate queries from Natural Language

## Introduction

Text to SQL is a preliminary study on the ability of using Machine Learning (ML) to translate natural language into SQL.

We learned the SQL language at BrainStation as part of the program. One of the first assignments was a series of questions around a database for which answers were to be found using SQL. An example was: *How many total trips were done in 2016?*

The knowledge of the SQL language to query database is essential to get valuable insights from the dataset. The purpose of this study is to figure out whether we can implement an algorithm that, given the question above, implements  the right query and returns the right result. This would be of great help to people that don't know SQL or they are just getting started.

## The Data

The task is not new to the ML community and few datasets exist out there to choose from. For this capstone project the wikiSQL dataset was chosen because of its large size (56,000 questions and SQL queries over 24,000 tables scraped from Wikipedia) and its helpful encoding. Each row included the English question, the table id and the encoding of the SQL query as a dictionary which key-value pairs were:

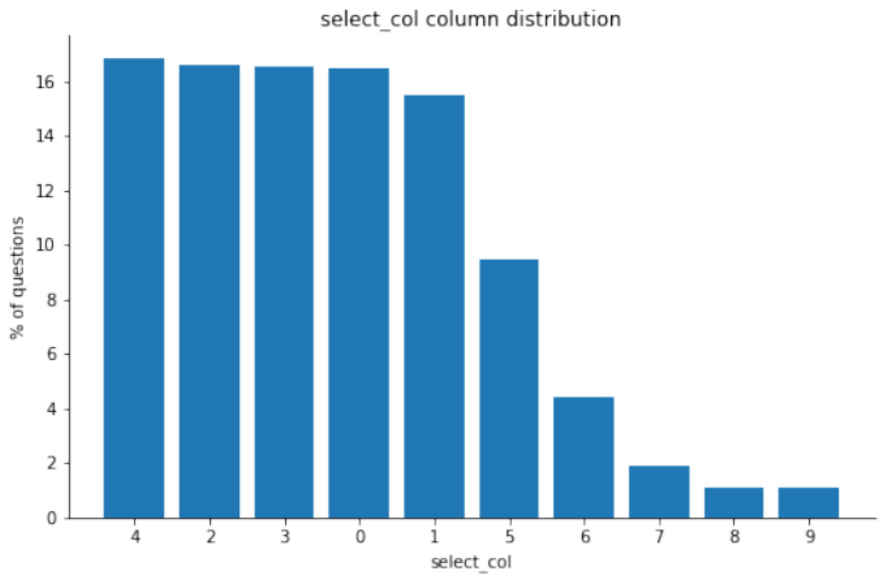
- `sel`: (SELECT) the numerical index of the column that is being selected
- `agg`: the numerical index of the aggregation operator that is being used
- `conds`: (WHERE) a list of triplets `(column_index, operator_index, condition)` where:

- `column_index`: the numerical index of the condition column that is being used
- `operator_index`: the numerical index of the condition operator that is being used
- `condition`: the comparison value for the condition, in either `string` or `float` type

## Data Cleaning and NLP

The data, provided as a `.json` file was parsed and loaded into a pandas dataframe through a customized python class. Preprocessing included cleaning and simplification (class reduction, `condition` was approximated to its base entity and encoded as number, multiple WHERE statements were eliminated and queries with no WHERE conditions were dropped). The final result of the pre-processing was a datafile for the *question* (features) and a datafile for the five query components (targets), five multi-class outputs (multi-class multi-output classification). All the five outputs suffered from class imbalance (figure below shows imbalance on `sel` as an example).

Using NLP, the *question*s were vectorized/tokenized in order to have a 2D matrix to feed into ML models.



## Modeling

Three multi-output classifiers were trained:

- Classifier Chain with SVC estimator. The estimator was chosen after exhaustive search of estimators (Logistic Regression, SVC, Random Forest and MultinomialNB) and hyperparameter optimization using GreadSearchCV
- Recurrent Neural Network (RNN) base model
- RNN with class weights to address class imbalance

Model performances were evaluated and compared.

Given the class imbalance, accuracy was not a good metric for the problem. Confusion matrices were actually the best way to gage the model performance. Adding class weights improved the RNN model true prediction rates.

The predictions were also decoded and the query execution accuracies (the number of queries that resulted in the correct result) were evaluated. Below is the summary table. Classification Chain scored the highest (5.8%).

| **Model** | **Execution accuracy** |
|---|---|
| Classification Chain | 5.8% |
| Base RNN | 5.6% |
| Weighted RNN | 3.0% |

## Conclusions

Below is the summary of findings:

- The problem was oversimplified because of its complexity
- Classifier Chain is a good predictor, especially because of its simplicity (out of the box model) compared to a more complex Deep Learning model
- The class imbalance is a problem that needs to be addressed with class weights
- The best approach for the problem could be a particular type of seq2seq model.