Improving Primary Key Detection with Machine Learning

Exposé

Janek Prange
February 21, 2022

1 Motivation

Big collections of data and therefore databases are becoming increasingly important as more data is produced with accelerating speed. In the best case, this data is directly saved in a format that contains a lot of metadata such as data types, primary keys and information about dependencies between columns.

In many scenarios however, it is not possible to design a proper schema suited for specific data beforehand. Instead, the data is already present in an arbitrary format or is being taken from other sources such as the internet, where the data format can rarely be influenced. These use cases require tools to – as much as possible automatically – detect the needed metadata from the raw data.

The thesis introduced in this exposé tries to propose a new solution to this problem making use of machine learning.

2 Existing Techniques

One of the main characteristics of a primary key is the uniqueness of its values. Based on this constraint, a naive way of finding primary key candidates would be to search for columns or column combinations which do not have any duplicate values. This could be achieved for example by sorting or hashing.

One existing algorithm which uses this characteristic is GORDIAN[1]. It works by organizing the data into a prefix tree, finding the largest column combinations that have duplicate values (maximal non-uniques) and from those compute the minimal column combinations which exclusively contain unique values (minimal uniques).

This technique has the disadvantage that the work increases with the size of the column combinations with duplicate values and is therefore not suited for large databases.

Another algorithm which tries to solve the problem of primary key detection efficiently is DUCC[2]. It has a vastly higher efficiency than GORDIAN, even though it detects both minimal uniques and maximal non-uniques at the same

[1] Yannis Sismanis, Paul Brown, Peter Haas, and Berthold Reinwald. Gordian: efficient and scalable discovery of composite keys. In VLDB, pages 691–702, January 2006.

[2] Arvid Heise, Jorge-Arnulfo Quiané-Ruiz, Ziawasch Abedjan, Anja Jentzsch, and Felix Naumann. Scalable discovery of unique column combinations. Proc. VLDB Endow., 7(4):301–312, December 2013. ISSN: 2150-8097. DOI: 10.14778/2732240.2732248. URL: https://doi.org/10.14778/2732240.2732248.

time. This is possible primarily because of its focus on aggressively pruning column combinations which can not be part of the desired solution.

3 Proposed Solution

The thesis introduced here will try to increase the efficiency of finding primary key candidates using a neural network which will be trained on a large set of tables. The hope is that the network will discover primary key candidates with better efficiency than the solutions described in Section 2.

In addition to the usage of neural networks, there may be other ideas to increase the efficiency of the algorithm. One possibility would be to sample from the columns of the table instead of reading all values. This way, the data that has to be retrieved from the database could be reduced, which may result in a significant speedup when the algorithm is used with large datasets.

Another opportunity for increased efficiency could be to narrow the selection of possible primary key candidates with filters. To begin with, columns which can not practicably be primary keys will be eliminated, such as columns containing images, videos or long texts. As a later filter, the neural network would try to detect the best primary key candidates from the remaining columns.

To reduce the scope of the thesis, the focus of the algorithm will be primarily on detecting single column primary key candidates. If the results are promising and there is time left, multicolumn candidates will be included.

4 Evaluation Method and Metric

To train the neural network that is at the center of the solution proposed in Section 3, methods have to be defined to evaluate the success or failure of the network. In this case, there are two favorable and relatively easy to measure metrics, which are the correctness and the efficiency of the algorithm.

A primary key candidate is correct when it has no duplicate elements and no subset of it is a column combination only consisting of unique values. These requirements are easy to check, especially if the algorithm is limited to single column candidates.

For the purposes of this thesis, the algorithm is efficient if it is faster and uses less main memory than the naive solution presented in Section 2, which will be tested on the same data as the neural network.