Predicting Garbage Collector Invocation

Team Name: _No__One_

1) Prajjwal Jaiswal

2) Manisha Meena

Problem

- Predict when Garbage Collection gets triggered

- Predict amount of free memory after each query

Insights

- ➔ Whenever Initial Free Memory goes down, Garbage Collector (GC) invokes
- ➔ Memory usage per unit time is almost constant for a particular query
- ➔ Whenever GC invokes, Free Memory goes up and Used Memory goes down

Insights

- ➔ When GC does not invoke, initial memory of particular instance would be final memory of previous instance

- ➔ When GC invokes, initial memory of particular instance would not be same as final memory of previous instance.

Solution

- ➔ Creation of Auxiliary data
- ➔ Feature Engineering
- ➔ Model to predict when GC invokes
- ➔ Prediction of GC trigger and free memory after each query
- ➔ Updation of Features if GC invokes

Auxiliary Data

- ➔ 91 unique queries obtained from training data
- ➔ Cases where GC invokes are excluded
- ➔ Memory used for each query is calculated as (Initial free memory - final free memory)
- ➔ Ratio of memory used and time taken by CPU is calculated
- ➔ Ratio does not vary much for instances with same query
- ➔ Final Auxiliary data contains 91 tokens and their corresponding mean of calculated ratios.

Feature Engineering

Features used in the prediction of GC invocation

- Initial free memory
- Initial used memory
- Mean_rate
- CPU time taken
- Estimated final used memory
- Estimated final free memory
- Ratio of initial free to total memory
- Ratio of estimated final free to total memory

For each query:

Memory Used = Final Used Memory - Initial Used Memory

Rate = Memory used / CPU Time Taken

Mean rate = Sum of Rates / Number of instances

Other features-

Estimated Final Used Memory = Initial Used Memory + Mean rate of executed query * CPU Time Taken

Estimated Final Free Memory = Initial Free Memory - Mean rate of executed query * CPU Time Taken

Proportion of Initial Free memory = Initial Free Memory / Total Memory

Proportion of Final Free memory = Final Free Memory / Total Memory

Model

Problems

1) **Imbalanced Data :**

   2259 cases where GC doesn’t invoke compared to just 171 cases where GC invokes.

2) **Errors possible in features :**

   Most built features are dependent on mean rate and not on the actual rate.

Model

Cure

Extreme Gradient Boosting

➔ **Ensemble learning**
Combination of weak learners to form a strong one.

➔ **Scale_pos_weight**
Deals with the imbalance in classes of Outcome Variable.

Predictions

- ➔ Given information of first instance is used to determine other features for the same instance and these features are then in turn used to calculate the features for next instance
- ➔ Triggering of GC is predicted by the whole set of features for a particular instance
- ➔ Final Used Memory and Final Free Memory get updated for the instance where GC invocation is predicted.

Limitations

Assumptions

- Initial Memory after every GC invocation attains the same value.
- Memory Usage per unit time is taken constant for each query token

Shortcomings

- Amount of memory cleaned by Garbage collector is not modelled
- Change in total memory is not monitored

Bugs

- ➔ Model will fail to predict if a new type of query is introduced

Bug Fixing

- ➔ Check if the asked query belongs to the initial token set
- ➔ If not found, assign it overall mean rate from training data instead of looking for its mean rate in auxiliary data

Proposed Alternate Model:

- ➔ Discard the assumption that rate for each query token is constant (when gc is not invoked)
- ➔ Build regression model to calculate Final used memory after execution of each query

Handling GC invocation

- ➔ Build a regression model to predict final used memory after GC has been invoked
- ➔ Build regression model to predict total memory after GC has been invoked
- ➔ Find final free memory from difference of total memory and final used memory
- ➔ RMSE decreased drastically
- ➔ Shortcoming of constant total memory assumption is resolved.

Thank you