SAPIENZA
UNIVERSITÀ DI ROMA

Visual Analytics Tool for AI Company logs

Visual Analytics

Professor:
Giuseppe Santucci

Student:
Giovanni Nicola
Della Pelle

Academic Year 2023/2024

Contents

1 Introduction

The project is based on a fictional scenario in which this tool is used.

1.1 Scenario

Our company provides AI tools for finance brokers. Its business scheme is based on a subscription model. Users of the company services can use the company website to analyze market trends using 4 different tools.

Each usage of a tool consists of a session, initialized by a request. This request includes user-specified parameters on the AI model underneath the tool. The user can generate a different response multiple tmes and has to rate the quality of the last response on a 1-5 scale.

The AI models elaborate in-house stored data to provide a response to the user. When some of the required input data are missing, a request to an external AI service is sent. In this case, the user request must wait until the missing data is received from the external service.

The company has received numerous complaints from customers about service quality. Major complaints regard output quality and loading time of the services. The company logs user sessions and requests to external services, and company engineers have provided guidelines on which parameters to inspect to identify the cause of the problems.

1.2 Intended User

The tool will be used by company engineers to inspect the causes of the problems. Its intended use is to provide live visualization of the company systems' logs. Its use can be extended in the future (modifying the server that provides the data) to show live and on-demand analytics.

1.3 Data Structure

The company tracks the following logs:

- User sessions
- Requests to the external AI services

User session log items have the following schema:

- timestamp
- User ID
- Model name/version

- Request parameters (parameters change depending on the model used)
- Session relevations
  - Satisfaction (user rating)
  - Generations (time the user generates the answer before accepting one)
  - Request tokens
  - WorkLoad Index (workload of the servers)

External service requests have the following schema:

- timestamp
- input tokens (tokens of the request)
- total tokens
- stream messages (number of messages in the service output)
- input dimension (dimension in MB of the input in the models that accept a non-textual input)
- loading time (round-trip time to receive a complete response)

The data covers one month of logs, approximately 20000 logs for user sessions and 6000 logs for service requests. The AS index results very high (270000 approximately)

2 Design

2.1 Requirements Analysis / Data Analysis

Company engineers have provided guidelines on which parameters to inspect to identify the cause of the problems. For the satisfaction/generations problem, we want to identify correlations between low satisfaction/high generations and the following parameters to inspect:

- number of tokens
  - domain: 3000 - 10000
  - type: integer
- WorkLoad Index (wli)
  - domain: 1 - 5
  - type: integer
- temperature (only for some models)
  - domain: 0.0 - 1-0
  - type: float
- presence penalty (only for some models)
  - domain: 0.0 - 2.0
  - type: float

The satisfaction parameter has a domain of [0, 12], while the satisfaction parameter is on a [1, 5] domain. The engineers also want to filter data by model, in order to see if one of them is underperforming.

For the high loading time problem, we want to identify correlations between loading time and:

- Input Tokens
  - domain: 1000 - 10000
  - type: integer
- Total Tokens
  - domain: 10000 - 80000
  - type: integer
- Stream Messages

  - domain: 1 - 12
  - type: integer

- Input Dimension (MB) (only for some requests)
  - domain: 0 - 8000
  - type: integer

- Weekday / Hour

2.2 Generations/Satisfaction Problem - Solution Design

To analyze the generations/satisfaction problem, the following graphs have been designed:

2.2.1 Scatterplot + dimension

TASK:

We want to visualize trivariate data:

- Satisfaction/generations - quantitative
- Parameters to inspect - quantitative
- Occurencies count
  - Both satisfaction and generations parameters have a small domain and are integers scales. In order to maintain a realistic view of the situation, the count data has been introduced, counting the occurrences of each tuple ¡satisfaction/generation, parameter to inspect¿

SOLUTION:

Scatterplot + dimension

- Y Axis: Generations/Satisfaction
- X Axis: Parameter to inspect
- Volume of the dot: occurrences count

Additionally, a filter is available to choose which model to visualize (or all of them).

2.2.2 WLI Boxplot

TASK:

We want to inspect in detail the correlation between Generations/Satisfaction and the wli parameter.

SOLUTION:

Boxplot:

- Y Axis: Generations/Satisfaction
- X Axis: wli

2.2.3 Tokens boxplot

TASK:

We want to inspect in detail the differences between the various models with respect to the tokens parameter.

SOLUTION:

Boxplot:

- Y Axis: Tokens
- X Axis: Generations/Satisfaction

The user can choose to visualize 4 charts, one for each model, or a single chart with all models.

2.3 Loading Time Problem - Solution Design

To analyze the loading time problem, the following graphs have been designed:

2.3.1 Scatterplot + Principal Component Analysis

TASK:

We want to inspect the correlation between loading time and the suspected parameters:

- Input Tokens - quantitative
- Total Tokens - quantitative
- Stream Messages - quantitative
- Input Dimension - quantitative

SOLUTION:

The solution consists in visualize, both at the same time, a scatterplot and a Principal Component Analysis plot. Given the resulting scatterplot, we think that visualize a projection using PCA could help in discovering insights.

Scatterplot

- Y Axis: Loading time
- X Axis: Parameter to inspect

2.3.2 Scatterplots Collection

TASK:

We want to inspect the correlation between loading time and weekday / hour of the day. The goal is to be able to compare different weekdays in order to spot patterns in the correlation weekday / high loading time. Another objective is to spot particular weekday / daytime tuples in which loading time is higher than average.

SOLUTION:

The solution consists of visualizing 7 scatterplots, one for each day of the week.

Scatterplot

- Y Axis: Loading time
- X Axis: Daytime

2.3.3 Linechart

TASK:

We want to inspect the correlation between loading time and weekday / hour of the day. The goal is to be able to compare different weekdays in order to spot patterns in the correlation weekday / high loading time. Another objective is to spot particular weekday / daytime tuples in which loading time is higher than average.

SOLUTION:

The solution consists of visualizing a linechart with 7 functions, one for each day of the week, color-coded.

Scatterplot

- Y Axis: Loading time
- X Axis: Daytime

2.4 Technology Tools

The tool consists of an Angular Single Page Application(SPA). The user can navigate the application using a top-bar menu that handles routing duties. The plots are realized with D3.js (scatterplots) and ObservableHQ Plot (linecharts and boxplots), with dedicated plot factories and customizable plot specs. All the graphics components (buttons, selects...) are from the Angular Material library. The backend is a NodeJS REST server connected to an InfluxDB database. It performs some basic computations for the visualizations (like the counting of equal tuples for scatterplot + dimension plots and the PCA calculations). It is not involved in the boxplot calculations, as they are performed by the frontend d3 library.

3 Related Works

3.1 Gatherplots

In *Deokgun Park et al.*, the authors encounter a challenge similar to the one related to one of our scatterplots. The problem is the fact that scatterplots suffer from overplotting when categorical variables are mapped to one or two axes.

To solve this probelm, the authors propose gatherplots, an extension of scatterplots to manage the overplotting problem. Gatherplots are a form of unit visualization, which avoid aggregation and maintain the identity of individual objects to ease visual perception. In gatherplots, every visual mark that maps to the same position coalesces to form a packed entity, thereby making it easier to see the overview of data groupings.

To achieve this result, the authors propose the following steps:

1. The Gather transformation, that non-linearly segments the graphical axis C and organizes data points in each segment to eliminate overplotting.

2. The choice of one of three layouts (Normalized, Absolute, SteamGraph). Each one of those layouts arranges stacked groups size and aspect rateo differently. The choice must be based on the type of data and axis.

Our dataset and chosen visualizations present a similar challenge in the Loading Time scatterplot, when the "stream messages" option is selected for the X axis. Here, differently from the other available options, the X-axis is non-continuous. A simple scatterplot would surely overplot and not be more useful than a boxplot. Our solution consists in:

- Adding a new variable that counts the occurrences of each tuple [x, y]

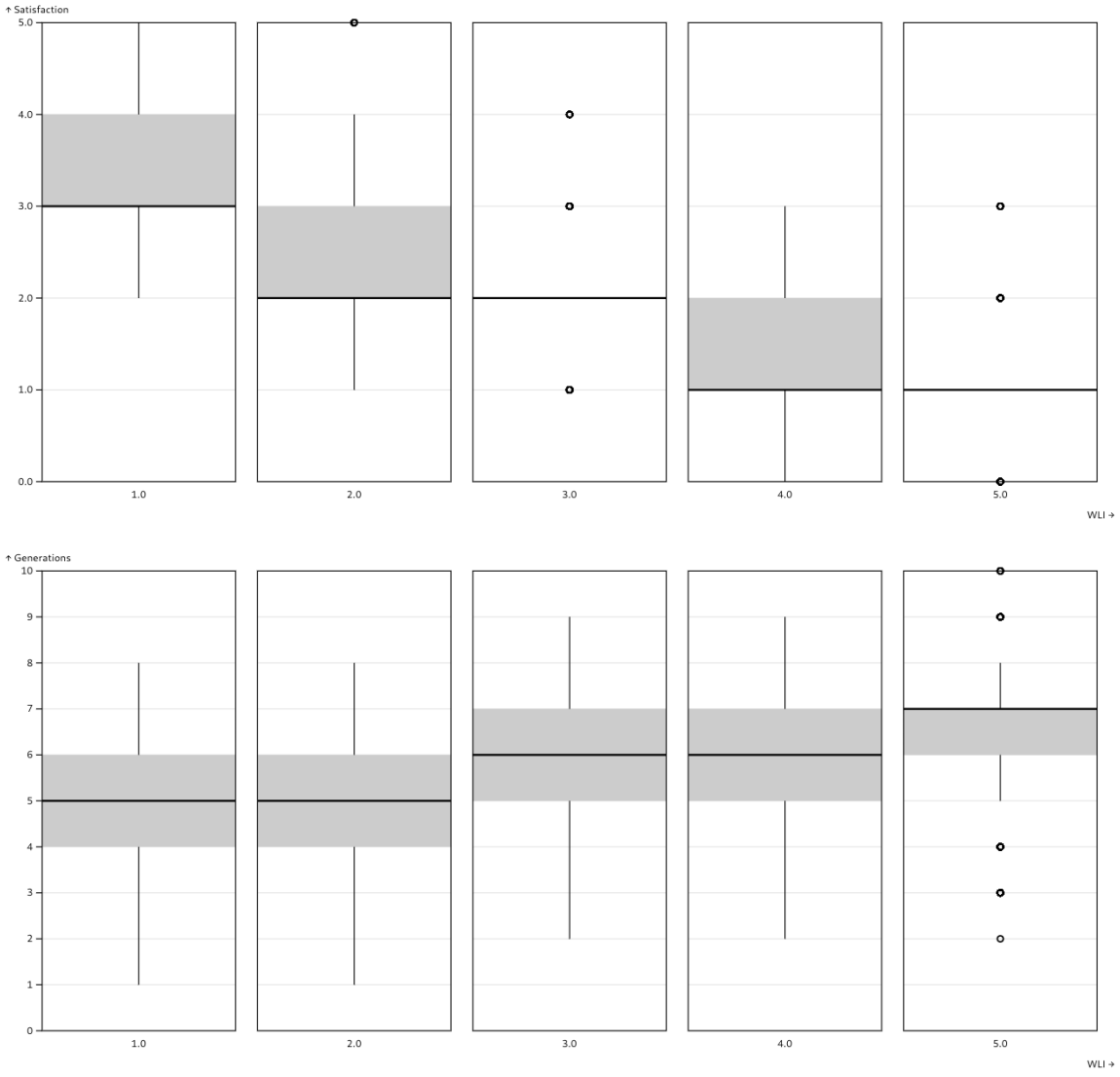
- Visualize the trivariate data as a scatterplot + dimension

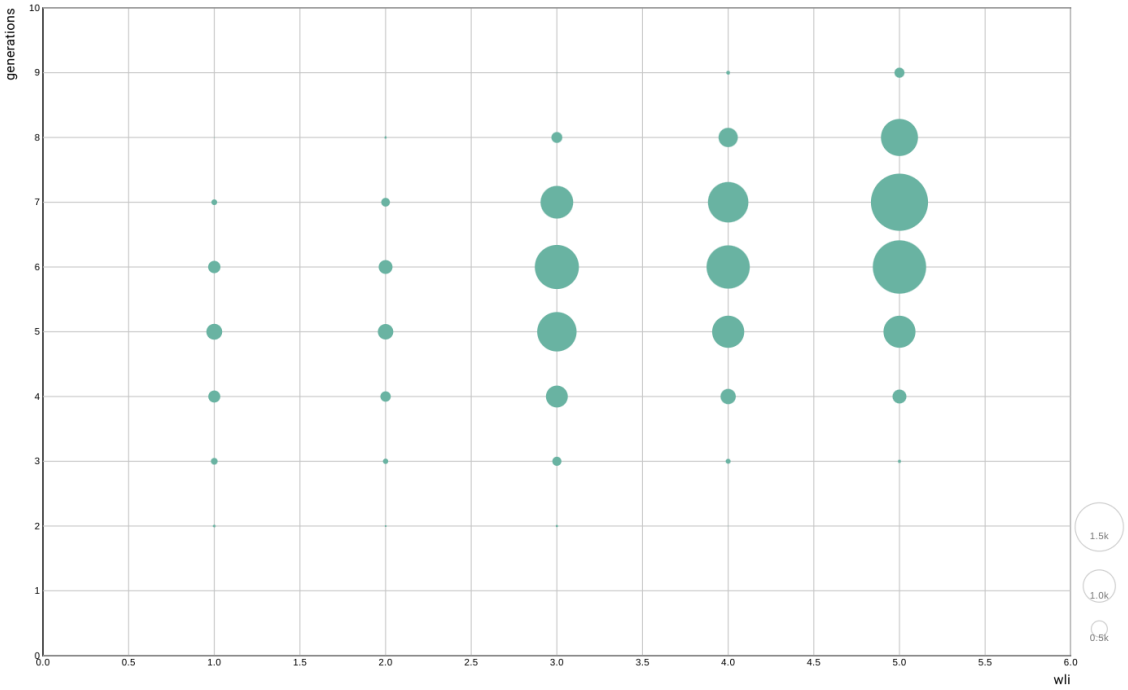
4 Discovered Insights

4.1 Satisfaction/Generations problem

4.1.1 WorkLoad Index

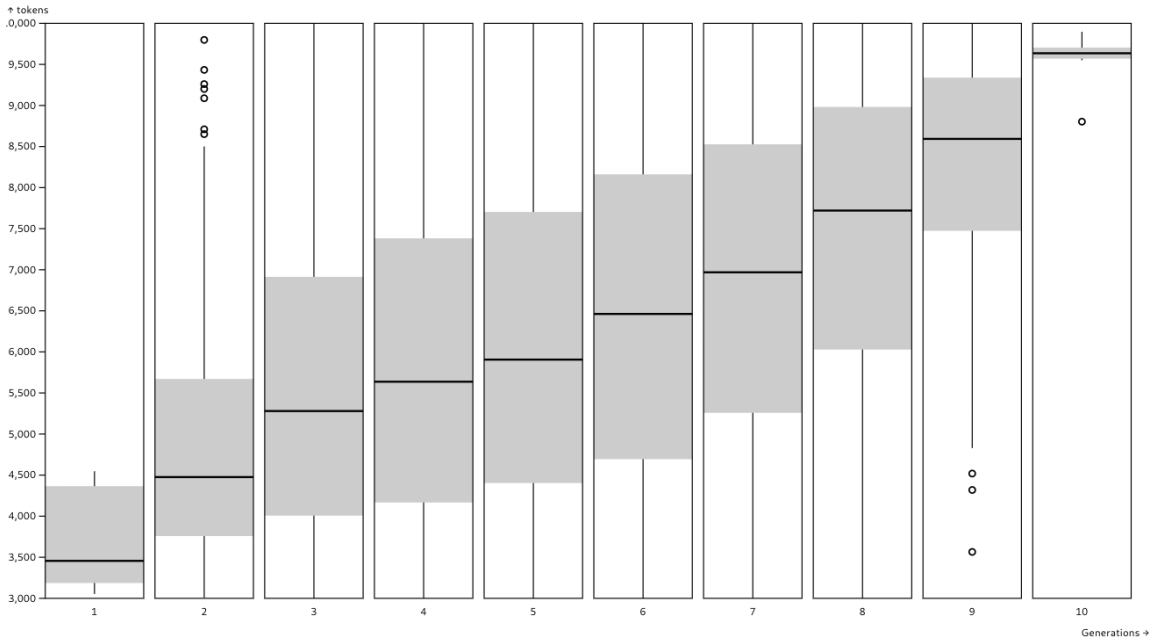
As shown by the two boxplots and the scatterplot, high WorkLoad Index decreases user satisfaction. The inverse correlation between satisfaction and generations is clear, with very high generations under heavy load.

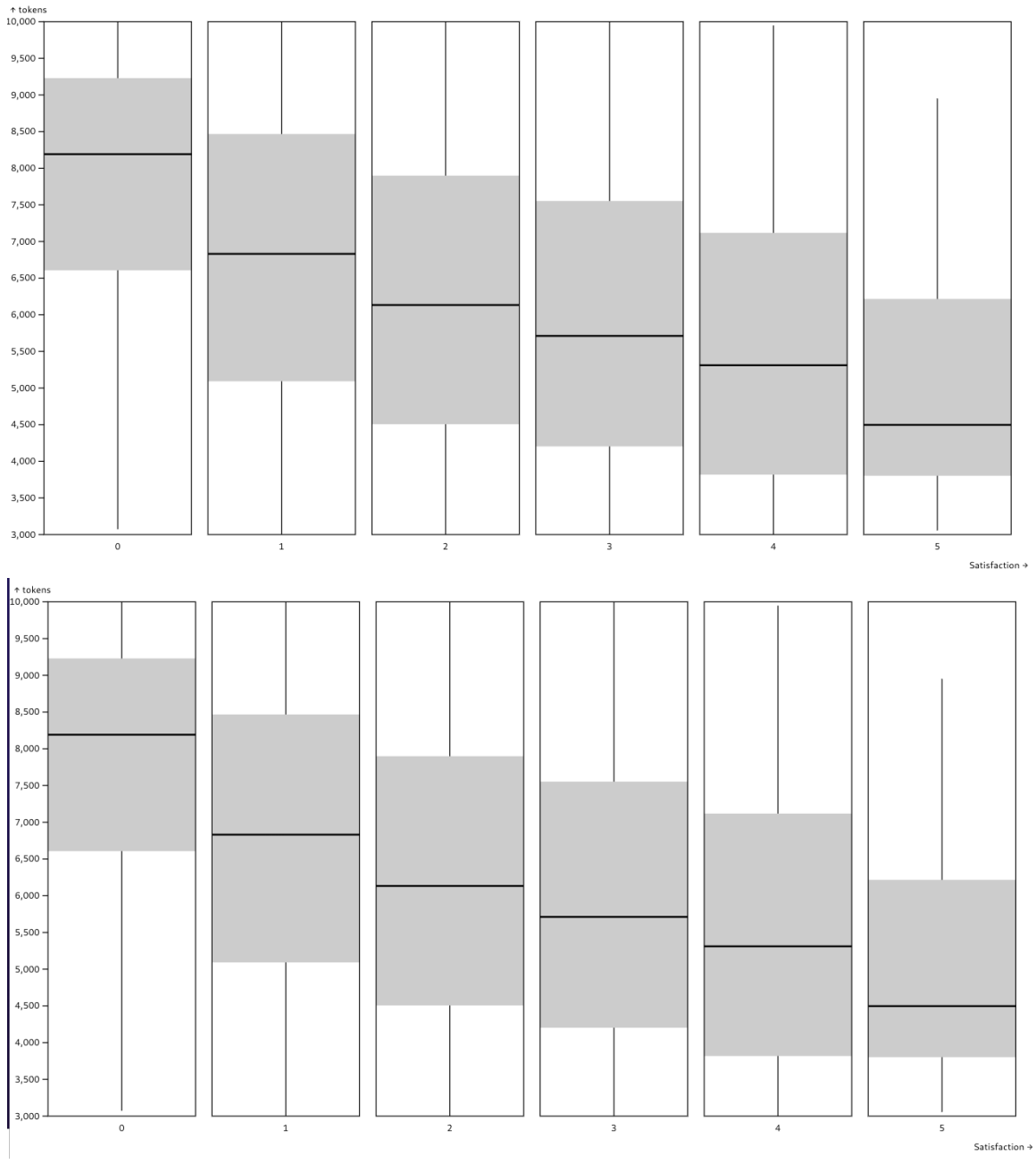




4.1.2 Tokens

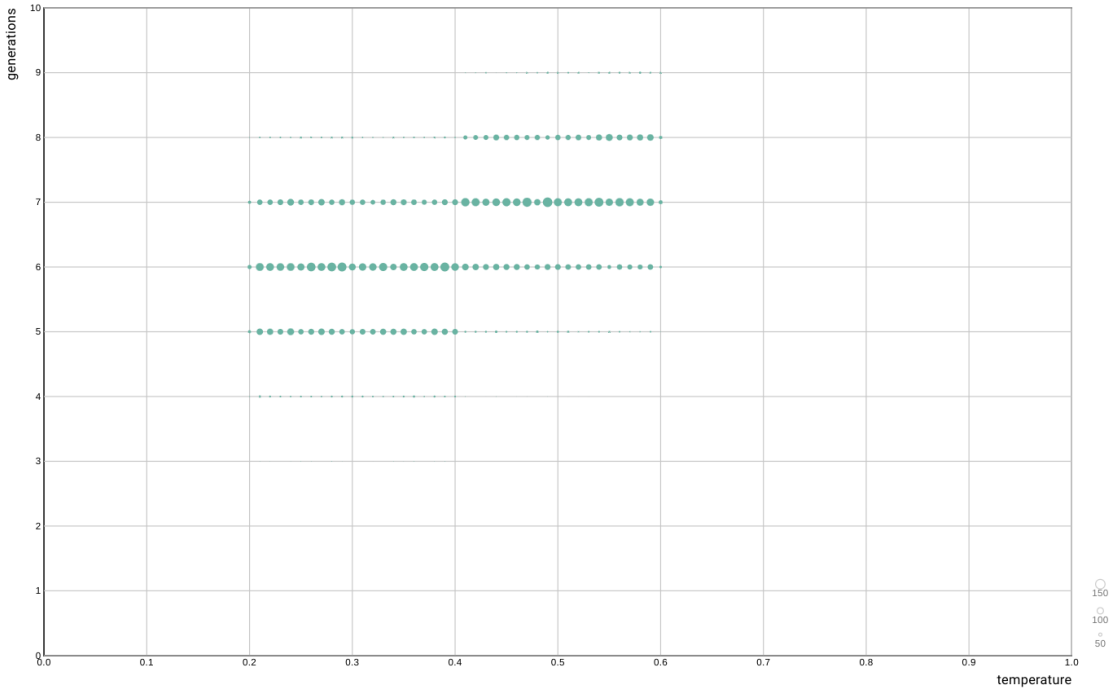
The same result can be seen for the tokens parameter. Higher tokens directly worsen user satisfaction.





4.1.3 Temperature

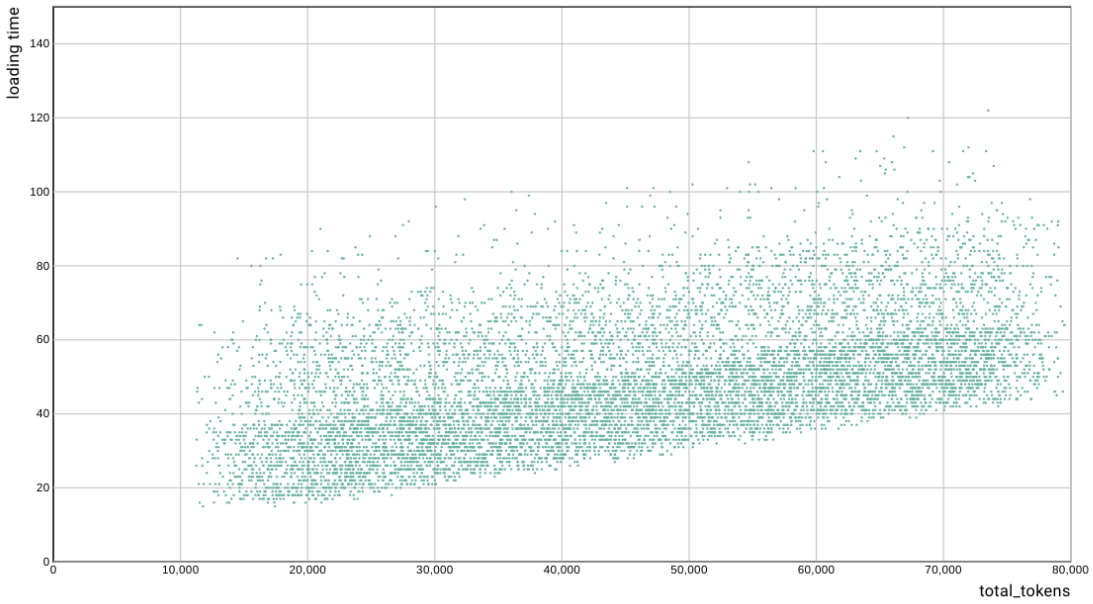
The graph shows how higher temperature deteriorate user satisfaction. Another noticeable insight is the preference of the users for a temperature between 0.2 and 0.7, while the temperature range is 0.0 - 1.0.

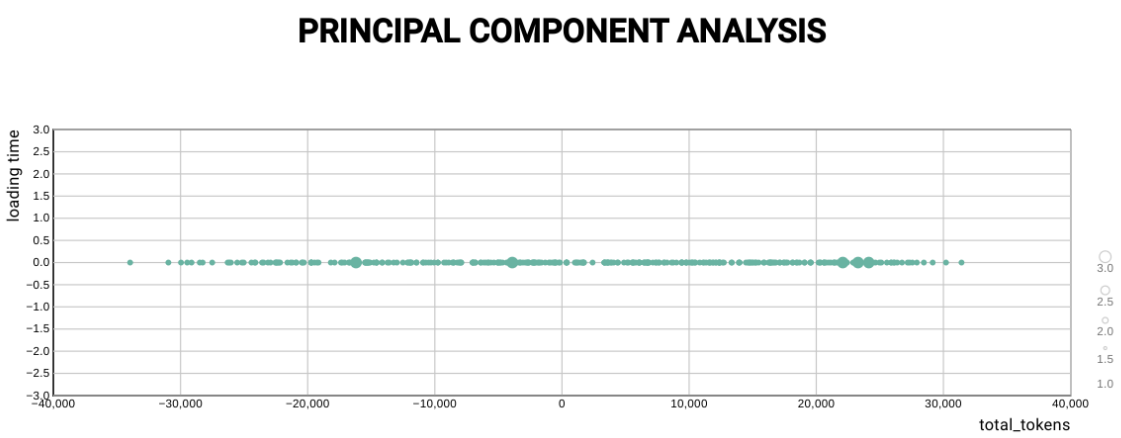

generations
10
9
8
7
6
5
4
3
2
1
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
temperature
150
100
50

4.2 Loading Time problem

4.2.1 Total Tokens

Higher total tokens directly increase loading time. The PCA doesn't give any insight.



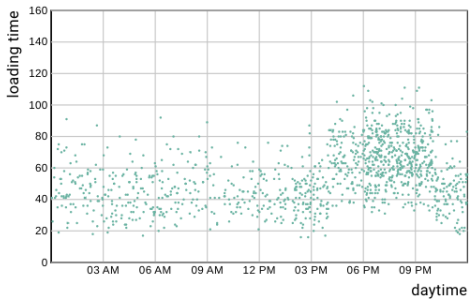


4.2.2 Weekday and Hour

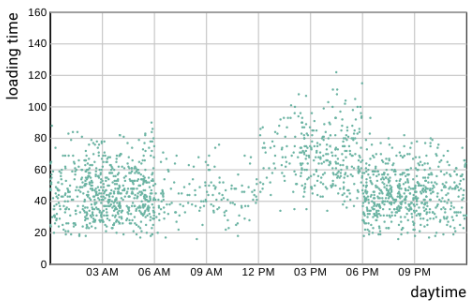
There are 3 main periods in which loading time is sensibly higher:

- Friday, between 16:00 and 22:00
- Saturday, between 12:00 and 18:00
- Sunday, between 22:00 and 24:00
- Monday, between 00:00 and 02:00

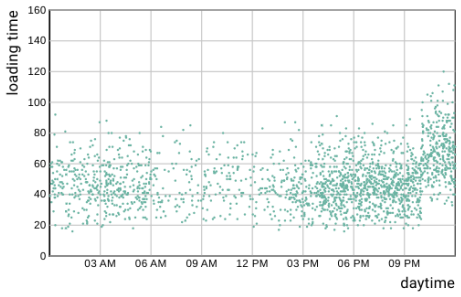
Friday



Saturday



Sunday



Monday