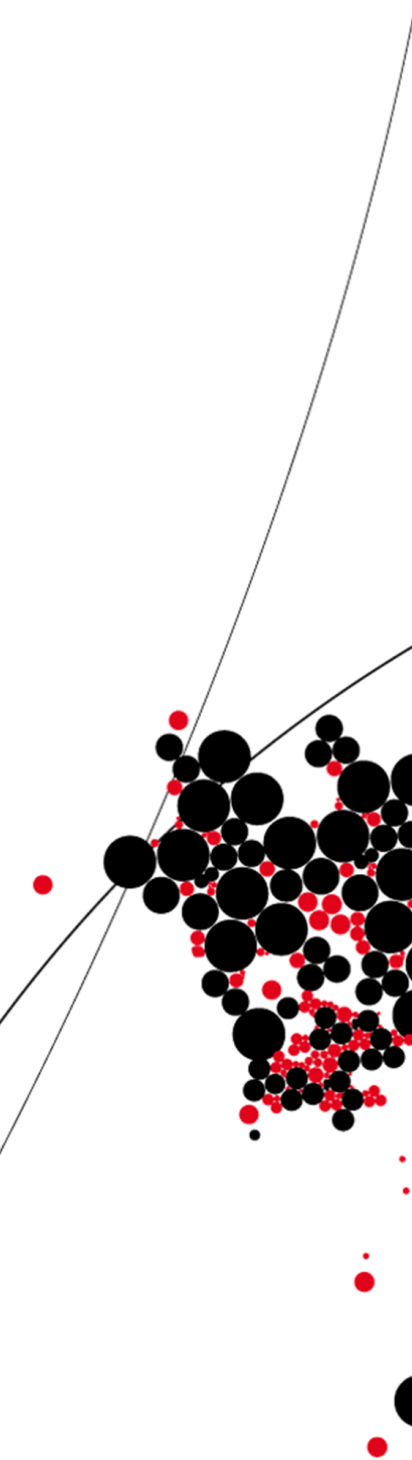




**UNIVERSITY OF TWENTE.**

**Faculty of Electrical Engineering,  
Mathematics & Computer Science**



**Improving usability at BetterBe  
through API analysis**

**Jarik G. Karsten**  
**M.Sc. Thesis**  
**July 2022**

---

**Supervisors:**

dr. M. Poel  
dr. P. van den Bos  
prof. dr. M.I.A. Stoelinga  
P. van Rossum

FMT & DMB  
Faculty of Electrical Engineering,  
Mathematics and Computer Science  
University of Twente  
P.O. Box 217  
7500 AE Enschede  
The Netherlands

---



# Contents

|   |            |
|---|------------|
| <b>List of acronyms</b>                 | <b>vii</b> |
| <b>1 Introduction</b>                   | <b>1</b>   |
| 1.1 Motivation . . . . .                | 2          |
| 1.2 Framework . . . . .                 | 2          |
| 1.3 Research questions . . . . .        | 3          |
| 1.4 Report organization . . . . .       | 3          |
| <b>2 Background</b>                     | <b>5</b>   |
| 2.1 Distance measures . . . . .         | 5          |
| 2.2 Evaluation metrics . . . . .        | 6          |
| 2.2.1 WCSS . . . . .                    | 6          |
| 2.2.2 Silhouette Coefficient . . . . .  | 6          |
| 2.2.3 Calinski-Harabasz Index . . . . . | 7          |
| 2.3 Clustering algorithms . . . . .     | 7          |
| 2.4 Rolling hash . . . . .              | 9          |
| 2.4.1 Representation . . . . .          | 9          |
| 2.4.2 Preprocessing . . . . .           | 9          |
| 2.4.3 Operations . . . . .              | 9          |
| <b>3 Related Work</b>                   | <b>13</b>  |
| 3.1 Use cases . . . . .                 | 14         |
| 3.1.1 Integrated analytics . . . . .    | 14         |
| 3.1.2 Workflow mining . . . . .         | 14         |
| 3.1.3 Clustering . . . . .              | 15         |
| 3.2 Usability . . . . .                 | 15         |
| <b>4 LSM Application</b>                | <b>17</b>  |
| 4.1 Frontend application . . . . .      | 17         |
| 4.2 Backend application . . . . .       | 21         |
| 4.2.1 Permissions/Tasks . . . . .       | 21         |

|          |   |           |
|----------|---|-----------|
| <b>5</b> | <b>Methodology</b>                          | <b>23</b> |
| 5.1      | Data Processing . . . . .                   | 23        |
| 5.1.1    | Kibana . . . . .                            | 23        |
| 5.1.2    | Data extraction . . . . .                   | 23        |
| 5.1.3    | Data format . . . . .                       | 24        |
| 5.1.4    | Data preprocessing . . . . .                | 24        |
| 5.2      | Data processing . . . . .                   | 26        |
| 5.2.1    | Tokenization of a workflow . . . . .        | 26        |
| 5.2.2    | Vectorization of documents . . . . .        | 27        |
| 5.2.3    | Vectorization of tasks . . . . .            | 28        |
| 5.3      | Clustering setup . . . . .                  | 29        |
| 5.4      | Clustering . . . . .                        | 29        |
| 5.5      | Analysis . . . . .                          | 30        |
| 5.5.1    | Centroids . . . . .                         | 30        |
| 5.5.2    | Task division . . . . .                     | 30        |
| 5.5.3    | Role distribution . . . . .                 | 31        |
| 5.5.4    | Page journeys . . . . .                     | 31        |
| 5.5.5    | Report generation . . . . .                 | 32        |
| 5.6      | Usability . . . . .                         | 32        |
| 5.6.1    | Interview . . . . .                         | 32        |
| 5.6.2    | Common workflows . . . . .                  | 32        |
| 5.6.3    | Common repetitions . . . . .                | 33        |
| <b>6</b> | <b>Data Processing</b>                      | <b>35</b> |
| 6.1      | Data preprocessing . . . . .                | 35        |
| 6.1.1    | Sessions and workflows . . . . .            | 35        |
| 6.1.2    | Whole sessions . . . . .                    | 35        |
| 6.1.3    | Human users . . . . .                       | 37        |
| 6.1.4    | Irrelevant calls . . . . .                  | 37        |
| 6.2      | Tokenization of a workflow . . . . .        | 38        |
| 6.2.1    | URIs . . . . .                              | 38        |
| 6.2.2    | Identifiers . . . . .                       | 39        |
| 6.3      | Vectorization . . . . .                     | 39        |
| <b>7</b> | <b>Determining Optimal Clustering Setup</b> | <b>41</b> |
| 7.1      | Distance measure . . . . .                  | 41        |
| 7.2      | Clustering algorithms . . . . .             | 41        |
| 7.2.1    | Fuzzy C Means . . . . .                     | 41        |
| 7.2.2    | Agglomerative . . . . .                     | 45        |
| 7.3      | Optimal setup . . . . .                     | 47        |

|   |           |
|---|-----------|
| <b>8 Clustering</b>                           | <b>49</b> |
| 8.1 Clustering . . . . .                      | 49        |
| 8.2 Evaluation . . . . .                      | 49        |
| 8.3 Cluster meanings . . . . .                | 52        |
| 8.3.1 Centroids . . . . .                     | 52        |
| 8.3.2 Task division . . . . .                 | 52        |
| 8.3.3 Role distribution . . . . .             | 53        |
| <b>9 Client comparison obstacles</b>          | <b>55</b> |
| 9.1 Limitations . . . . .                     | 55        |
| 9.2 Removal of parameters . . . . .           | 55        |
| 9.3 Task based approach . . . . .             | 57        |
| 9.4 Outlier removal . . . . .                 | 58        |
| 9.5 Summary . . . . .                         | 58        |
| <b>10 Page Journeys</b>                       | <b>61</b> |
| 10.1 Determining page journeys . . . . .      | 61        |
| 10.2 Rolling hash . . . . .                   | 62        |
| 10.2.1 Representation . . . . .               | 62        |
| 10.2.2 Repetitions . . . . .                  | 62        |
| 10.2.3 Filtering results . . . . .            | 63        |
| 10.3 Results . . . . .                        | 63        |
| 10.3.1 Configure, Search, Quotation . . . . . | 63        |
| 10.3.2 Person, Quotation . . . . .            | 63        |
| 10.3.3 Lol, Table . . . . .                   | 64        |
| 10.3.4 Vehicle, Table . . . . .               | 64        |
| <b>11 Report Generation</b>                   | <b>65</b> |
| 11.1 Generation . . . . .                     | 65        |
| 11.1.1 Steps . . . . .                        | 65        |
| 11.1.2 Invocation . . . . .                   | 66        |
| 11.2 Report . . . . .                         | 67        |
| <b>12 Usability</b>                           | <b>69</b> |
| 12.1 Documentation . . . . .                  | 69        |
| 12.2 Training . . . . .                       | 70        |
| 12.3 Design . . . . .                         | 70        |
| 12.3.1 Vehicle, Table . . . . .               | 70        |
| 12.3.2 Lol, Table . . . . .                   | 71        |
| <b>13 Discussion</b>                          | <b>73</b> |

|                            |           |
|----------------------------|-----------|
| <b>14 Future Work</b>      | <b>75</b> |
| <b>15 Conclusion</b>       | <b>77</b> |
| <b>References</b>          | <b>79</b> |
| <b>Appendices</b>          |           |
| <b>A Tasks</b>             | <b>83</b> |
| <b>B Cluster Tasks</b>     | <b>87</b> |
| <b>C Interview</b>         | <b>89</b> |
| <b>D Report generation</b> | <b>93</b> |

# List of acronyms

**API**      Application Programming Interface

**DSL**      Domain-Specific Language

**GUI**      Graphical User Interface

**HCI**      Human Computer Interaction

**IDF**      Inverse Document Frequency

**LOL**      Lease Object Language

**LSM**      Lease Service Management

**REGEX**   Regular Expression

**TF**      Term Frequency

**TF-IDF**   Term Frequency - Inverse Document Frequency

**URI** Uniform Resource Identifier

**UUID** Universally Unique Identifier

**WCSS** Within-Cluster Sum of Squares



## Introduction

Application Programming Interface (API)s play a critical role in software development. They come in various forms and protocols. External APIs, or open APIs, are available to external users or companies. These can provide services for location & mapping, payment, search, etc. Examples include the APIs of Google Maps and Paypal. Internal APIs on the other hand are for use within the company or project. They are not available to external users and can be used to communicate various services or resources within the company. With micro-service architectures, an API gateway can process requests and redirect them to the appropriate internal API [1].

A recent SlashData survey in 2020 has concluded that nearly 90% of software developers use APIs. These software developers use 2.9 APIs on average of which 69% are external, and the remaining are internal. By using these external services construction time can be greatly improved.

To facilitate the use of external APIs, they have to be well-designed and easy to use. If this is not the case, a developer can opt for easier-to-use APIs or even write the service from scratch [2]. Additionally, in both internal and external APIs, complexity can cause longer implementation times, and possibly lead to bugs due to incorrect usage. A study from 2008 reported that API users were able to complete tasks 2.4 to 11.2 times quicker if the desired method was in the expected class [3].

Thus, due to its importance, usability has become an important factor in Human Computer Interaction (HCI) research [4]. It was found that flaws in the usability of APIs will have an impact on development time and correctness of the resulting software. Additionally, for companies offering external API services, usability is an important construct to retain customers. In this paper, we will be utilizing a dataset of BetterBe, consisting of manual API calls, and calls from their Graphical User Interface (GUI) directly to their API. BetterBe is a software company offering

services, including various APIs, to car leasing companies. We will propose an approach to identify common workflow patterns of the API and use these to improve the usability of the application at BetterBe. These workflows encompass multiple, sequential, related requests sent by a user.

## **1.1 Motivation**

Recently, BetterBe exposed a GUI to clients which was previously intended for internal use. It is currently being used by clients manually and automatically by receiving automated calls directly to the API. This prompted additional development into the system to make it better suited for clients, however, it was unclear exactly how the system was being used by their clients. Additionally, BetterBe is in the process of making its clients more self-sufficient in using the application.

The need to gain insights into how the system is used and the goal to increase the self-sufficiency of users motivates the need to identify use cases of their system. This allows us to determine common usage patterns or common workflow patterns within the application. These workflows consist of sequential actions by a user to achieve a certain task. They might perform multiple search requests in order to finally retrieve a certain item. These sequential, related requests can then be grouped together into a single workflow.

By identifying the most common workflows, development can be focused on these areas. Additionally, insights into the data can help customize client training. In the long term, these developments and insights should aim to reduce the time spent helping/training clients in using their application. Finally, by locating inefficient workflows that unnecessarily span multiple pages, we can restructure certain pages to improve these workflows to improve the usability of the GUI.

## **1.2 Framework**

As BetterBe has multiple services, they use tools such as ElasticSearch [5] to store and manage their API requests. Furthermore, they use Kibana [6] which serves as an extension to ElasticSearch to gain insights into stored data through queries, charts, and anomaly detection.

This thesis will continue the work of previous research in which an approach to improving usability of the API was detailed [7]. This research also provides a background into the data transformation, clustering techniques, and their evaluation in Chapter 2. The research will be conducted on BetterBe's Lease Service Management (LSM) application. Further information on the application will be provided in Chapter 4.

### **1.3 Research questions**

The approach will utilize previous usage data of the API to extract workflow patterns in the problem domain. These API requests will then be clustered to identify common workflow patterns between users. Finally, we will improve the usability of the BetterBe system. This leads to the following research questions.

1. How can we extract workflow patterns from gathered API usage data?
2. How can we identify common workflow patterns between users?
3. How can we improve the usability at BetterBe by analyzing the data and common workflow patterns?

### **1.4 Report organization**

The remainder of this report is organized as follows. First, we will start by introducing relevant background that we will use throughout the report. We will then provide previous work in Chapter 3, and continue by introducing the application of BetterBe in Chapter 4. The methodology is then discussed in Chapter 5 followed by the implementation details in Chapters 6, 7, 8. Chapter 9 will discuss approaches when comparing different clients. Chapter 10 investigates the paths users take through the website. The data from previous Chapters are incorporated into an automated pipeline resulting in a generated report which will be discussed in Chapter 11. Finally, usability will be discussed in Chapter 12 followed by the discussion, future work, and conclusion in the final chapters.



# Background

In this chapter, we will explain concepts and algorithms that we will use throughout the report.

## 2.1 Distance measures

We will use the Cosine and Jaccard similarity measures as a base for determining the distance between user sessions in the application once they have been converted to vectors. They are defined by the following equations:

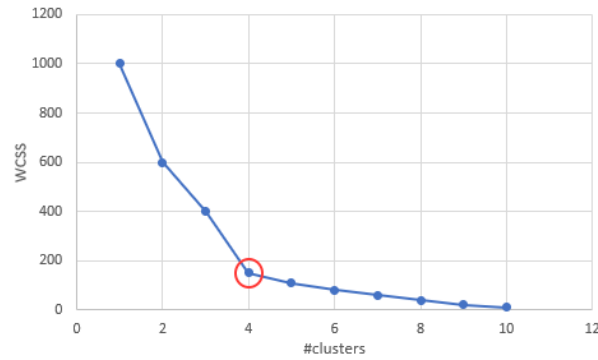
$$\text{cosine\_similarity} = \frac{X \cdot Y}{\|X\| * \|Y\|} \quad (2.1)$$

$$\text{jaccard\_similarity} = \frac{X \cdot Y}{\|X\|^2 + \|Y\|^2 - X \cdot Y} \quad (2.2)$$

They are both monotonic functions in the domain of  $[0, 1]$  when considering vectors that do not contain negative values. The largest similarity occurs at 1, so we represent them as  $\text{cosine} = 1 - \text{cosine\_similarity}$  and  $\text{jaccard} = 1 - \text{jaccard\_similarity}$  to have a distance measure.

The Cosine distance is independent of document length as it considers the angle between two vectors. Large documents result in higher values in the vector. If we multiply a vector by  $k$  it will simply extend the vector in the same direction. Thus, it will have a distance of 0 to the previous vector.

However, Jaccard does not have this feature. When multiplying a vector by  $k$  it will have a distance of  $1 - \frac{k}{1+k^2-k}$  to the previous vector. In case of  $k = 2$  this is already  $\frac{1}{3}$ .



**Figure 2.1:** Elbow method

## 2.2 Evaluation metrics

We will discuss the Within-Cluster Sum of Squares (WCSS), Silhouette Coefficient, and Calinski-Harabasz Index evaluation metrics which can be used to determine the number of clusters used in the clustering.

### 2.2.1 WCSS

The WCSS utilizes the elbow method and considers the variation within clusters. The elbow method plots multiple WCSS values for the different numbers of clusters and determines the optimal number of clusters by the elbow in this plot. This can be seen in Figure 2.1. We can observe that the WCSS decreases rapidly until it reaches 4 clusters. At this point, it starts decreasing at a slower rate and we get diminishing returns for increasing the number of clusters. Thus, an 'elbow' is created at 4 clusters making it the optimal amount.

The WCSS can be calculated using the following equation

$$WCSS = \sum_{c \in C} \sum_{d \in c} distance(centroid(c), d)^2 \quad (2.3)$$

where  $C$  is our set of clusters,  $centroid(c)$  provides the centroid for cluster  $c$ , and  $distance(x, y)$  provides the distance between two points based on the distance measure.

### 2.2.2 Silhouette Coefficient

The Silhouette Coefficient provides a value in the range of  $[-1, 1]$  where 1 indicates the clusters are clearly distinguished, and -1 shows that the clustering is likely wrong. It calculates the value for each data point and averages this value. Thus, we can

also look at the Silhouette Coefficient for each cluster.

It uses the following equation to calculate the Silhouette Coefficient

$$Silhouette\_Coefficient = \begin{cases} 1 - x/y & x < y \\ y/x - 1 & x > y \\ 0 & x = y \end{cases}$$

where  $x$  indicates the average distance of a vector  $v$  to its own cluster  $C$ , and  $y$  indicates the average distance to its closest neighbouring cluster. These are given by the following equations

$$x = \frac{\sum_{vector \in C} distance(v, vector)}{|C|} \quad (2.4)$$

$$y = \min_{cluster} \frac{\sum_{vector \in cluster} distance(v, vector)}{|cluster|} \quad (2.5)$$

### 2.2.3 Calinski-Harabasz Index

The Calinski-Harabasz index calculates the separation and cohesion of data points. The separation indicates the distance to other clusters, and cohesion distance to its own cluster. Let us denote  $N$  as the number of documents,  $K$  as the number of clusters,  $C_k$  as the centroid of cluster  $k$ ,  $C$  as the global centroid, and  $S_k$  as the size of cluster  $k$ . This results in the following equations.

$$separation = \frac{\sum_{k=1}^K S_k * \|C_k - C\|^2}{K - 1} \quad (2.6)$$

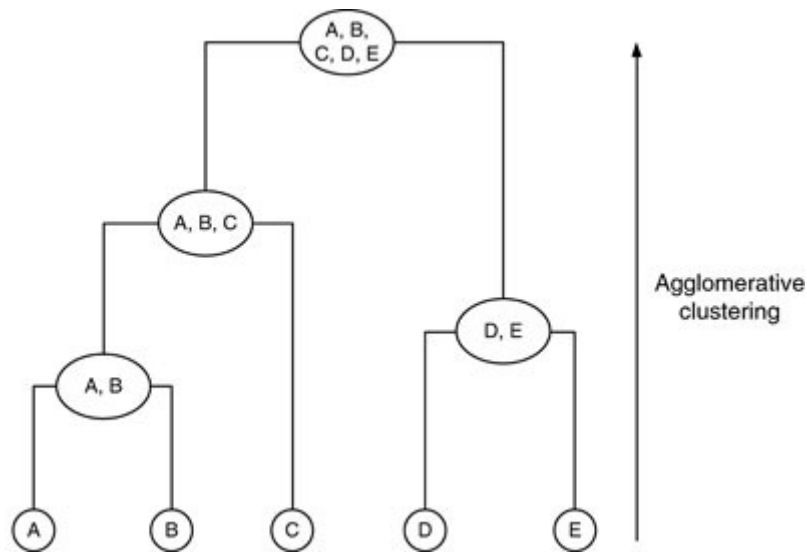
$$cohesion = \frac{\sum_{k=1}^K \sum_{d \in C_k} \|d - C_k\|^2}{N - K} \quad (2.7)$$

$$Calinski - Harabasz = separation/cohesion \quad (2.8)$$

## 2.3 Clustering algorithms

### Fuzzy C-means

The algorithm takes the number of clusters,  $k$ , as input from the user. It then initializes a coefficient  $\in [0, 1]$  for each  $(cluster, data\_point)$  pair. These  $k * n$  coefficients are initialized at random, where  $n$  indicates the total number of data points, and the



**Figure 2.2:** Agglomerative clustering [8]

highest coefficient of a data\_point can be used to determine the cluster. Each of these  $k$  clusters will have a centroid which is the average point out of all data\_points belonging to the cluster. This can be used to determine the error of each cluster by taking the distances of the data\_points to their centroid. The algorithm will update these coefficients in each iteration resulting in new centroids, and a lower error rate. This process is repeated until the maximum coefficient change falls beneath a threshold  $\epsilon$ .

A downside of Fuzzy C-means is that it will converge on a local minimum which does not have to be a global minimum. Thus, the solution depends on the initial coefficients.

### Agglomerative clustering

Agglomerative clustering or hierarchical clustering is initialized with a cluster for each data\_point. These clusters will subsequently be merged in a bottom-up fashion based on the lowest mean distance between a pair of clusters. This repeats until we have  $k$  clusters, defined by the user. This process can be seen in Figure 2.2, where we start with 5 data\_points that ultimately merge into a single cluster. If we were to specify 2 clusters, the process would terminate before the last merge.

A downside of the agglomerative clustering is the time complexity of  $O(N^3)$  where  $N$  is the total amount of data\_points to cluster. This is mainly due to the fact that each possible merging is considered at each step.



## 2.4 Rolling hash

When utilizing a rolling hash [9], an array of numbers can be represented as a polynomial with an associated hash. Integers can be added/removed from both sides in constant time ( $O(1)$ ) while simultaneously updating the hash. We will later use these to represent a list of pages to locate common user journeys through the application in Chapter 10. We will discuss the representation as a polynomial, preprocessing, and supported operations in the following sections.

### 2.4.1 Representation

An integer array  $C$  can be represented as

$$hash = C_0 * a^0 + C_1 * a^1 + C_2 * a^2 \dots C_n * a^n$$

where  $C_i$  indicates the integer at index  $i$  of the array, and  $a$  is the prime base. The hash is taken modulo  $m$ . We will take  $a = 22695477$  and  $m = 2^{32}$  as this is a common pair proven to have a low collision rate [10].

### 2.4.2 Preprocessing

To perform the operations in  $O(1)$ , we compute the modular inverse of  $a$  under  $m$ , and the powers  $a^i$ . The modular inverse of  $a$  under modulo  $m$  can easily be found in  $O(\log(m))$  using Euclid, which has been implemented, however, we can simply provide the value here as 690295837. The initialization of values and precomputations can be found in Algorithm 1. The Queue() is a data structure that supports adding/removing entries in the front and back of a list in  $O(1)$ . Java has the LinkedList implementation, while Python has the deque data structure.

### 2.4.3 Operations

#### Remove first integer

The first integer  $c$  of the array corresponds to the  $C_0 * a^0$  term in the polynomial. Thus, we can subtract  $c$  from the hash which we can obtain from our queue. We can then divide the entire polynomial by  $a$  to obtain the previous structure of powers ranging from  $a^0$  to  $a^{n-1}$ . As we are in a modulo field we obtain this by multiplying the hash by the inverse of  $a$  under modulo  $m$  which we previously precomputed. Finally, we perform the modulo operation on the  $hash$ . This process is described in Algorithm 2.

---

**Algorithm 1: Preprocessing**

---

**Input:** Maximum string size  $n$ **begin**

```
    /* Parameters */
    hash  $\leftarrow$  0 ;
    a  $\leftarrow$  22695477 ;
    m  $\leftarrow$   $2^{32}$  ;
    queue  $\leftarrow$  Queue() ;
    inversea  $\leftarrow$  690295837 ;
    /* Precompute powers */
    pow  $\leftarrow$  [1] * (n + 1) ;
    index  $\leftarrow$  1 ;
    while index  $\leq$  n do
        pow[index]  $\leftarrow$  (pow[index - 1] * a) mod m ;
        index  $\leftarrow$  index + 1
```

---

---

**Algorithm 2: Remove First**

---

**begin**

```
    c  $\leftarrow$  queue.popLeft() ;
    hash  $\leftarrow$  hash - c ;
    hash  $\leftarrow$  hash * inversea ;
    hash  $\leftarrow$  hash mod m
```

---

### Remove last integer

The last integer  $c$  of the array corresponds to the  $C_{n-1} * a^{n-1}$  term in the polynomial where  $n$  indicates the length of the array. The size of the array can be obtained from the queue in  $O(1)$ . Thus, we can subtract  $c * a^{n-1}$ , and perform the modulo. Note that these powers have been precomputed. This process is described in Algorithm 3.

---

**Algorithm 3: Remove Last**

---

```
begin  
   $n \leftarrow queue.size() - 1$  ;  
   $c \leftarrow queue.popRight()$  ;  
   $hash \leftarrow hash - c * pow[n]$  ;  
   $hash \leftarrow hash \bmod m$ 
```

---

### Add first integer

To append an integer  $c$  to the front of the array we can multiply the entire polynomial by  $a$  to increase the power of each entry. This frees up the  $C_0 * a^0$  term. Thus, we can simply add this integer  $c$ , and perform the modulo. This process is described in Algorithm 4.

---

**Algorithm 4: Add First**

---

```
Input: Integer  $c$   
begin  
   $hash \leftarrow hash * a$  ;  
   $hash \leftarrow hash + c$  ;  
   $hash \leftarrow hash \bmod m$  ;  
   $queue.addLeft(c)$ 
```

---

### Add last integer

To append an integer  $c$  to the end of the array we have to add the term  $c * a^n$  to the polynomial. We perform the multiplication using the precomputed values of  $a^i$ , and perform the modulo. This process is described in Algorithm 5.

---

**Algorithm 5: Add Last**

---

**Input:** Integer  $c$ **begin**

```
 $n \leftarrow \text{queue.size}();$   
 $\text{hash} \leftarrow \text{hash} + c * \text{pow}[n];$   
 $\text{hash} \leftarrow \text{hash} \bmod m;$   
 $\text{queue.addRight}(c)$ 
```

---

# Related Work

As the web matured and saw more use, usability research started to focus on this environment. This led to the creation of design principles relating to layout [11], navigation structure [12], searching [12], etc. As the literature advanced, various studies have defined usability factors/guidelines including, but not limited to: easy to learn [13], consistency [14], and easy to interpret client code [13]. These have subsequently been used to evaluate APIs in surveys, controlled experiments, and task-based usability tests [13]–[19].

Though these manual approaches can provide valuable insights into usability, among clients at BetterBe there are large differences in tasks being performed. Thus, an automated approach is preferred. This can be achieved by utilizing existing data on each client.

Studies that did not involve users often used existing bug reports, gathered programmer discussions, or analyzed code changes over time [13], [20]–[22]. As an example, [13] categorized bug reports into usability factors including missing features (43.5%), correctness (31.1%), documentation (27.3%) etc.

Though technology can change significantly over time due to development, these changes often do not disrupt the basic principles of usability, leaving them mostly unchanged [23]. [24] is still often cited, and describes this usability as 5 factors [25]:

1. ease of learning how to use the application
2. efficiency of the application design in terms of fast navigation and required clicks
3. ease to memorize how to use the application.
4. low number of errors made by users.

## 5. general satisfaction of the user with the application

In this paper, we will mainly focus on factors (1), (2), and (4). To improve upon the user interactions with the application, we will first research how the application is currently being used. We will then discuss how this knowledge can be leveraged to improve usability.

## **3.1 Use cases**

To detect use cases, or workflows, of users, there are multiple approaches discussed in the literature.

### **3.1.1 Integrated analytics**

Firstly, analytics can be integrated into the website to track user behavior. [26] discusses the possibility of integrating Google Analytics into the website. A 'dashboard' can then be built that tracks users and provides insights into common resources, and the number of clicks required to reach this resource. Additionally, it provides insights into where users drop out of the workflow to reach these resources.

After design changes have been made to counterattack the discovered problems, the results can be easily observed in the dashboard after which the process can be repeated.

The main drawbacks include the manual encoding of tracking labels as this has to be done for each link on the website. Otherwise, the complete user journey through the website cannot be tracked. Additionally, after implementing this, the system has to be monitored, and thus it can take significant time before gaining results.

### **3.1.2 Workflow mining**

To automate the process of finding workflows, existing usage data can be mined to find workflows. An algorithm and approach to this problem is proposed in [27]. It explores the possibility of viewing sessions as lists of services being utilized with temporal relations between the services. This temporal relation includes an ordering and time spent in each of the services. These services can be considered as pages, or sets of pages and actions. This data can be mined from logged API data as this often includes time, Uniform Resource Identifier (URI), and originating page.

Common patterns can then be located between the various sessions.

The approach can be useful in finding common actions between users but faces some issues when dealing with the similarity between sessions. This is mainly due to the fact that additional 'noise', in the form of pages, can be inserted in sessions making it difficult to find larger actions as services no longer exactly match. Thus, it is less suitable for similarities between whole sessions, but in Chapter 10 we explore the approach to locate common patterns within sessions.

### **3.1.3 Clustering**

In [28] the use of clustering to group user sessions with similar usage patterns together was proposed. This approach would use previous usage data of users to cluster their sessions within the application to analyze usage patterns on the website to subsequently recommend website restructuring changes. The process can be automated, and 'noise' within sessions which could disrupt the approach taken in Section 3.1.2 has a smaller impact and will likely not affect the classification of such a session.

## **3.2 Usability**

After we extract workflow patterns from API usage data and identify common workflow patterns between users, we can use these to improve usability.

In [20], problems relating to usability factors such as missing features (43.5%), correctness (31.1%), documentation (27.3%), etc. could be mainly categorized as a lack of understanding in the upward and downward direction. The upward direction, focused towards the problem domain, is described as "mapping desired scenarios in the problem domain to the content of the API". The downward direction focuses on an understanding of the API and behavior of the system. Code snippets for workflows are preferred to the documentation of individual methods. Thus, by utilizing these common workflow patterns, the documentation can be tailored toward these workflows. Additionally, training can also focus on these areas. This will improve the understanding in the upward and downward direction of the application to subsequently improve ease of learning (1) and reduce the number of errors (4).

Additionally, we can analyze these common usage patterns to propose design changes in the application. These can improve the application through navigation and feature improvements. This will benefit the efficiency of the interface design (2).





# LSM Application

In this chapter, we will provide background on the LSM application. The LSM application facilitates leasing cars by storing data on cars to support operations related to searches, quotations, and price calculations. First, we will introduce the frontend application used by customers. Secondly, we will provide relevant details related to their backend.

## 4.1 Frontend application

Initially, we start on the page seen in Figure 4.1. Users can search for cars, and configure settings in the menu on the left. After selecting a certain car they go to the screen seen in Figure 4.2. It provides an overview of the car with additional configurations that can be added. The left menu shows required choices such as exterior/interior color. The items in the right list provide additional features to the car.

Once the required choices have been made, a quotation can be created as seen in Figure 4.3. After submitting the quotation we get a brief overview, but can request additional vehicle details. This can be seen in Figures 4.4 and 4.6. Finally, a user can also download a pdf of the car here which can be seen in Figure 4.5.

To support these various price calculations, BetterBe developed Lease Object Language (LOL) which is a Domain-Specific Language (DSL). The goal of LOL is to support various calculations such as utilizing data stored on cars to determine a lease price and cost for a car. By tailoring their DSL to this end it becomes easier for a non-programmer to make changes in these scripts. While BetterBe does provide a template, these LOL scripts are further developed by clients so we will not show these. However, these scripts to determine lease prices depend on factors such as price per kilometer, vehicle price, option price, and residual value.

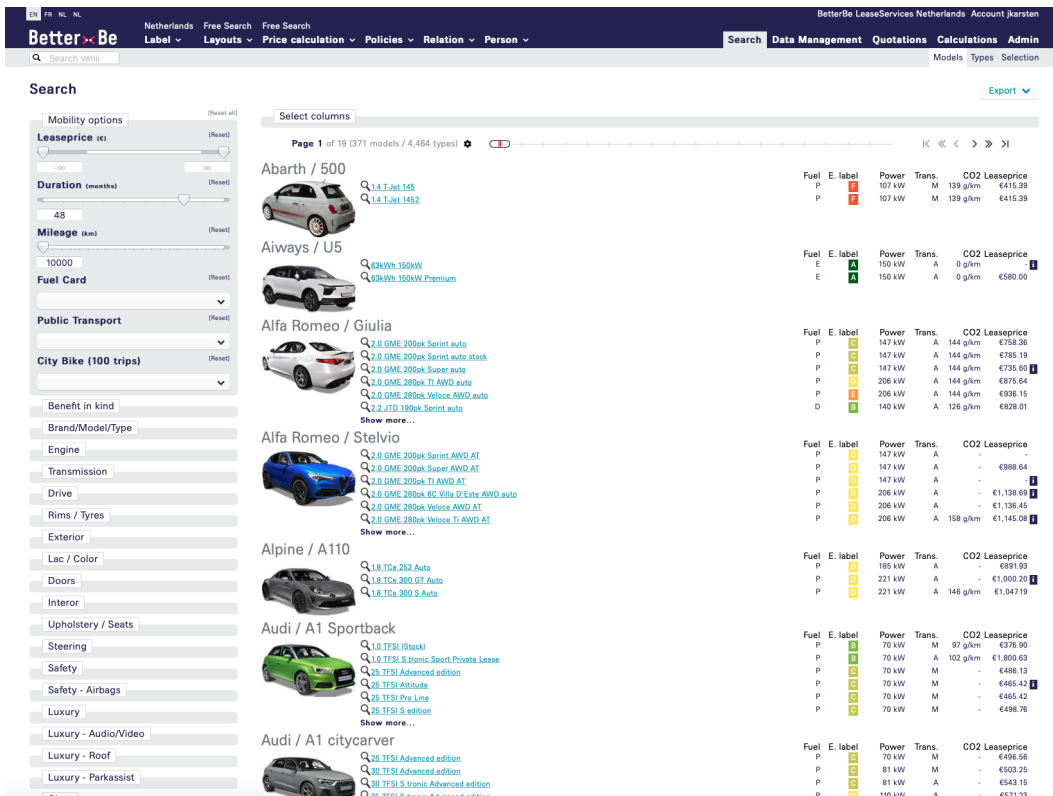


Figure 4.1: Search overview

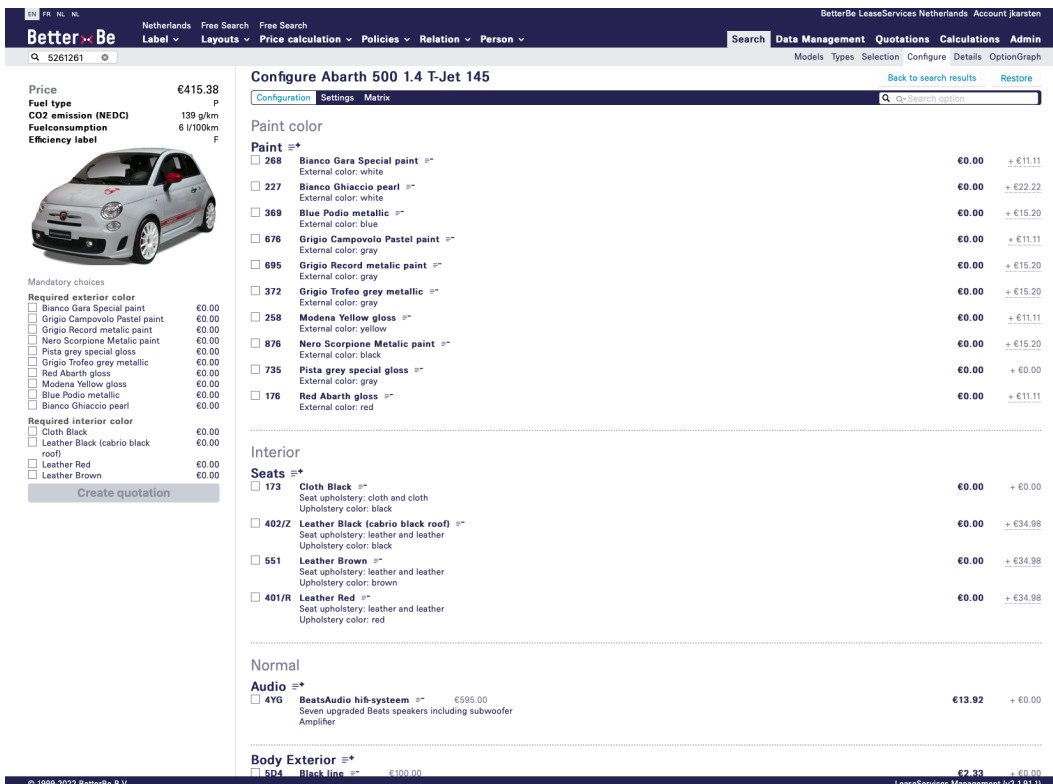


Figure 4.2: Car details

**Price** €426.50  
**Fuel type** P  
**CO2 emission (NEDC)** 139 g/km  
**Fuelconsumption** 6 l/100km  
**Efficiency label** F



**Required**  
 Exterior paint Bianco Gara, special pastel paint €11.11

**Current configuration**  
 Cloth Black €0.00  
 Bianco Gara Special paint €0.00

**Create quotation**

Figure 4.3: Create quotation

BetterBe LeaseServices Netherlands Account jkarsten  
 Search Data Management Quotations Calculations Admin  
 Quotations Calculations Templates Relations Persons

**Quotation 1156** [Calculation](#) [Vehicle details](#) [Configure vehicle](#) [default PDF](#) [Delete](#)  
 2022-03-30 11:11:16 2022-03-30 11:11:18

|                    |                                      |  |
|--------------------|--------------------------------------|--|
| quotationid (UUID) | 09807f0b-38bd-434f-8357-a77e964539bb |  |
| quotationid        | 1156                                 |  |
| calculationid      | 1541                                 |  |
| Price              | €426.50                              |  |

Calculation parameters

|                        |       | Unit   |
|------------------------|-------|--------|
| Tax bracket            | 2     |        |
| Duration               | 48    | months |
| Max contribution       | €0.00 |        |
| Contribution mandatory | N     |        |
| Mileage                | 10000 | km     |

Vehicle

|         |   |
|---------|---|
| Make    | Abarth  |
| Model   | 500   |
| Type    | 1.4 T-Jet 145   |
| Message | Failed to find public_transport record with code = null |

Vehicle Options

|  | Base price | Retail price |
|--|------------|--------------|
| Cloth Black                                      | €0.00      | €0.00        |
| Bianco Gara Special paint                        | €0.00      | €0.00        |
| Exterior paint Bianco Gara, special pastel paint | €393.00    | €475.00      |

© 1999-2022 BetterBe B.V. LeaseServices Management (v2.192)

Figure 4.4: Quotation overview

**Quotation number: 1166**

**Configured vehicle: Abarth 500  
1.4 T-Jet 145**

Brand: Abarth  
Model: 500  
Type: 1.4 T-Jet 145  
Fueltype: Petrol  
CO<sub>2</sub>-emission: 139 g/km

**Options:**

- Leather Black (cabrio black roof) € 0.00
- Black leather sport seats € 1,236.00
- Pista grey special gloss € 0.00

**Leaseprice: € 450.37**  
**Duration: 48 months**  
**Yearly mileage: 10,000**



**Figure 4.5: Quotation pdf**

The screenshot displays the 'Details' page for a vehicle with ID 5261261, an Abarth 500 1.4 T-Jet 145. The interface includes a top navigation bar with 'BetterBe' branding and various menu options like 'Label', 'Layouts', 'Price calculation', 'Policies', 'Relation', and 'Person'. A search bar is present on the right. Below the navigation, there are tabs for 'Specification', 'Summary', 'Equipment', 'Options', 'Paint & Interior', 'LPG', 'Parks', 'Fleet', 'Accessories', 'Configuration', and 'Resources'. The main content area is divided into several sections, each with expandable items:

- Version:** Make (ABARTH), Model (500), Year (2014), Type (1.4 T-Jet 145), Number of doors (3), Body type (hatchback), Vehicle type (Cars), JATO Leasing Code (7703214120120160601C), Type (combustion), Trim (-), Transmission type (manual).
- Charges:** percentage discount (2), Price (23510 €), Base price (13972), national tax 3 amount (6603.88), amount excluding VAT (520.66), Latest price list number (1 juni 2016).
- Dimensions:** overall length (mm) (3660), overall width (mm) (1627), overall height (mm) (1485), wheelbase (mm) (2300), rear seat up, to lower window (l) (165).
- Audio:** Speakers (standard).
- Fuel:** Fuel type (unleaded), capacity (l) (35), main (main).
- Engine:** cc (?), 1388; Liters (?), 1.4; number of cylinders (?), 4; number of valves per cylinder (?), 4; CO2 emission (NEDCI) (139 g/km), Efficiency label (F).
- Transmission:** Driven wheels (front), number of speeds (5).
- Performance:** Power (107 kW, 145 Hp), rpm for maximum power (low) (5500), Torque (210 Nm), rpm for maximum torque (low) (3000).
- Wheels:** tire load index (front: 84, rear: 84), type (front: conventional, rear: conventional), run flat (front: no, rear: no), Alloy rim (yes).
- Warranty & Service:** duration (months) (24), distance (km) (999999, 30000).
- test:** cruise control (-), adaptive cruise control (-), BBAirco (y), Electronic window control (no), Rain sensors (no), multi function steering wheel (yes), Navigational systems (-), includes phone (yes), Front parking sensors (no), Rear parking sensors (yes), radio (AM/FM), heated front seats (halogen), heated driver seat (no).
- Leaseparameters:** BBChannel (-).

At the bottom of the page, there is a footer with '© 1999-2022 BetterBe B.V.' and 'LeaseServices Management (v2.1.92)'.

Figure 4.6: Vehicle details

All these factors and more have to be defined. These might include vehicle prices, option prices, images and many more factors. These factors relating to price calculations are stored in calculation tables. In Figure 4.7 we can see a small example of data related to fuel prices. Data unrelated to price calculations such as images or reports are stored in the data management tab.

## 4.2 Backend application

When a request is made by a user, it is logged in ElasticSearch with all the relevant data. It is subsequently routed to a specific method in the LSM controller. These methods will return the required data or page to the user. Which methods a user can invoke depends on their permissions.

### 4.2.1 Permissions/Tasks

To figure out which permissions a user has, we have to check their roles and tasks. BetterBe has defined global tasks in its application. These tasks have a set of as-

Calculation table "brandstofprijs"

| ct_brandstofprijs_id | code | prijs  |
|----------------------|------|--------|
| 6                    | P    | 1,839  |
| 5                    | N    | 1,025  |
| 4                    | Hd   | 1,513  |
| 3                    | G    | 0,8491 |
| 2                    | E    | 0      |
| 1                    | D    | 1,513  |
| Add new row          |      | 2,1410 |

**Figure 4.7:** Fuel price data

sociated permissions. Clients can create roles with a certain set of tasks. Thus, customer-specific roles are created with a certain set of permissions.

Methods in the backend application require certain permissions to execute. For example, a quotation permission is required to generate quotations or download them in pdf format. If the user does not have this permission, this action will disappear from the frontend application. We will later use these permissions and their associated tasks to describe user behavior.

# Methodology

In Section 3.1.2, we discussed the possibility of using algorithms to detect common user actions through pattern analysis. However, there were limitations in dealing with similarities between whole sessions. To overcome these challenges we propose an approach that translates these sessions into a format that allows us to cluster sessions to determine similarities.

The clustering is split into 3 steps which we will discuss in this chapter: data processing, clustering setup, and clustering. In Chapter 10 we will then explore the approach of [27] in which common user actions are located within sessions. In Chapter 11, it will be discussed how these findings are summarized in an automated report. Chapter 12 will discuss the implications for usability.

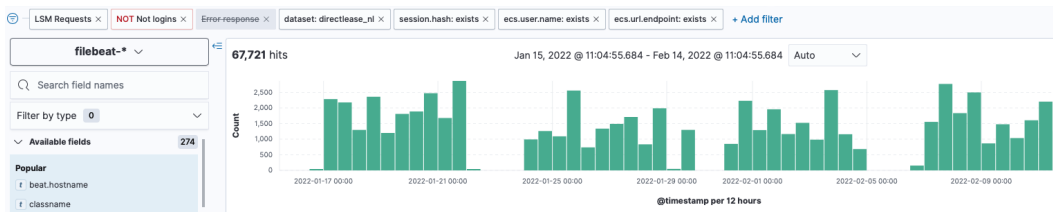
## 5.1 Data Processing

### 5.1.1 Kibana

In Chapter 4 we introduced the application developed by BetterBe. User actions in this application generate API calls. BetterBe primarily uses Kibana, an extension of Elasticsearch, to log these API requests. As seen in Figure 5.1, we can filter these requests as seen at the top. We filter for requests made when logged in by a specific customer "directlease\_nl". Furthermore, we ensure that relevant fields exist to remove spurious requests.

### 5.1.2 Data extraction

Extracting data by querying Kibana is limited to a total of 10.000 hits or results. Direct extraction from Kibana is possible in the form of a CSV file, but this is limited



**Figure 5.1:** Filtered requests to the application in Kibana.

to 10MB. Increasing this limit is experimental and untested as it might put too much stress on the cluster. This makes these approaches unsuitable for large data sets.

Libraries can be utilized to simplify data retrieval, but due to security reasons, this could not be set up for the original server. Thus, a second server was set up with the relevant data. This allowed automatic data retrieval through library interfaces that connect to the Kibana server.

### 5.1.3 Data format

The data from the Kibana response is provided in JSON. It contains the following relevant parameters for each call:

1. Timestamp (iso-format): When the request was sent by the user.
2. Session hash (string): Session hash of the user.
3. Username (string): User sending the request.
4. User role ([string]): Array of roles associated to the user.
5. Dataset (string): Name of the dataset being queried. In the form of <customer>\_<country> or <customer>\_<location>.
6. Endpoint (string): Class and method that will process the API call.
7. API endpoint (string): Path of the user request along with parameters.
8. Call type (string): GET, POST, PUT or DELETE.
9. HTTP referer (string): URI where the request originated from.

### 5.1.4 Data preprocessing

When processing the data we will first cluster the calls into the sessions. A session will contain all the calls after a user logs in until the user logs out. We can then filter



for complete sessions from human users. Finally, we will remove irrelevant calls. These processes are described in the following sections.

## **Sessions and workflows**

In the data related to a call, a session hash is tracked. Thus, we can group the calls by username and session hash to obtain a session. A session should consist of a single workflow, which consists of a process by a user utilizing an arbitrary amount of related, sequential API calls with a certain goal. This might entail the process of searching for a car, and creating a quotation.

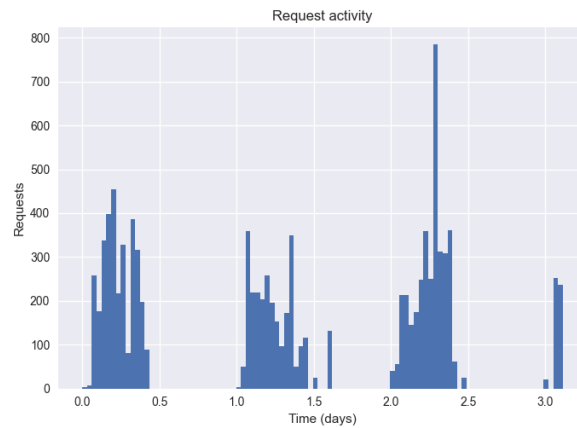
We will have to account for the fact that a user might perform multiple workflows within a single session. This will limit the correctness of the clustering as a single cluster should not contain multiple workflows. This can be inferred from analyzing the average number of unique requests per session length. To determine this we do not have to search through the requests URIs as this would be tedious due to varying/optional parameters. Rather, we can look at the method (endpoint) handling the request as there is a one-to-one mapping of requests to backend methods. This plot of unique calls for each session length should not increase drastically as the session length increases. If there is no significant increase in unique activity for longer sessions these do not have to be removed. This will be investigated in Section 6.1.1, but can differ when investigating data from other sources.

## **Filter incomplete sessions**

We want to avoid cutting off a session before it has been completed as this might create outliers within clusters or even new clusters. Thus, we want to remove these incomplete sessions. To achieve this, additional data outside the timeframe of the dataset can be gathered. If a session's hash occurs outside the timeframe of the dataset it can be removed. Alternatively, as requests are made by human users, we can have downtime in the requests during nighttime. This can be seen in Figure 5.2 where request activity is shown over a period of 3 days. This allows us to perform a split on the downtime.

## **Human users**

Certain clients may run automated calls. These might originate from test scripts or automated pieces of client code. As we are clustering workflows from users, we will



**Figure 5.2:** Downtime in the requests over 3 days of request data

remove these outliers. This will be done by plotting the average number of requests per minute of sessions and determining a cutoff point.

### Irrelevant calls

From each of the sessions, we can remove irrelevant calls. These have to be manually determined for each dataset. In the BetterBe environment, each page has several images. These can correspond to cars that were retrieved by a search or simply a car being displayed. A separate 'resource' call is made for each of these images. This can result in 20 consecutive resource calls if a large search was performed. These resource calls were removed.

## 5.2 Data processing

After we have a set of workflows  $W$ , we translate these into Term Frequency - Inverse Document Frequency (TF-IDF) vectors. This requires us to convert our workflows to a set of words/strings, also known as tokens. We can then create our vectors.

### 5.2.1 Tokenization of a workflow

#### Determining relevant fields

Timestamp, session hash, and username were used for determining the workflow and are not required in the representation of a workflow, and will not be taken into account as tokens. The dataset parameter simply indicates which data the request

originates from. This will be the same for each workflow so it does not have to be taken into account. This leaves us with user roles, endpoint, API endpoint, and call type.

## Workflow representation

Finally, for each of our workflows  $w$ , we will generate a single document  $d$  containing all tokens generated by the calls made within the workflow. The union of all unique tokens will be our dictionary of terms  $T$ . We can now translate these documents to a numerical vector representation.

### 5.2.2 Vectorization of documents

As discussed, we currently have a set of documents  $D$  corresponding to our workflows. We can now translate these to numerical format. In areas of document clustering, document querying, and text mining, documents are translated into a numerical format. TF-IDF vectors have often been used, and many extensions exist in specific areas [29]–[31]. A TF-IDF vector assigns a value to each term in every document based on how much information they contain. Low-frequency words will contain more information than a word that occurs in every document. By doing this for every  $(term, document)$  pair, we will have a vector representing each document.

Terms are valued proportionally to the number of times they appear in the document, offset by the number of times it is found in the entire document set. First, we calculate the Term Frequency (TF),  $tf(t, d)$ , for a term  $t$  within a document  $d$ . This is done by

$$tf(t, d) = \frac{freq_{t,d}}{\sum_{t' \in D} freq_{t',d}} \quad (5.1)$$

where  $freq_{t,d}$  indicates the amount of times term  $t$  occurs in the document  $d$ . Secondly, we calculate the Inverse Document Frequency (IDF), which measures the amount of information a word contains. It will provide higher values to rare terms in the total dataset as opposed to a common term. It is given by the following formula

$$idf(t, d, D) = \log\left(\frac{N}{doc\_freq(t) + 1}\right) \quad (5.2)$$

where  $N$  is the number of documents and  $doc\_freq$  is the number of documents that term  $t$  occurs in. To avoid division by zero errors, we add one to the denominator. These metrics allow us to now calculate the TF-IDF values for each term in the document using

$$tf - idf(t, d, D) = tf(t, d) * idf(t, D) \quad (5.3)$$

This results in a structured vector  $X$  where each index,  $X_i \in \mathbb{R}^+$ , corresponds to a term from the complete dataset.

### 5.2.3 Vectorization of tasks

In the previous vectorization, parameters specific to clients are utilized. To allow comparison between clients or combining data from various clients, we will provide an alternative vector based on tasks being performed in a session. The clustering can then be performed similarly to the approach taken with the TF-IDF vectors.

#### Determining tasks

As there are many methods handling API requests, we can create scripts to search for the required permissions of each method through all the relevant files. As mentioned in Section 4.2.1, we discussed that these permissions can be linked to tasks. These tasks and permissions can be viewed in Appendix A. This results in the following mappings:

1.  $Method \rightarrow Permission^*$ : maps a method to a set of associated permissions.
2.  $Permission \rightarrow Task^*$ : maps a permission to a set of parent tasks.

A permission often only has one task that is it part of with a few exceptions. In these cases, we cannot clearly determine which task was being performed. We will then take all tasks which is 2-3 in our case.

#### Determining the vector

At this point, we have determined the total set of tasks  $T$ . For each session, we will then count the number of times each of these tasks  $t$  were performed. This can then translate into a vector containing these counts

$$vector = [count(t) | t \in T] \quad (5.4)$$

where  $count(t)$  indicates the number of calls in the session that are related to task  $t$ .

Finally, we will normalize this vector have all indices of the vector within a range of [0, 1]. For this we will use the following equation:

$$normalized = (value - min\_value) / (max\_value - min\_value) \quad (5.5)$$

where the `min_value` and `max_value` are the minimum and maximum values. We will do this at each index separately. If we were to divide by the total sum, indexes corresponding to rare tasks would all tend to zero. This approach will work for a Cosine distance measure due to its independence of document length, but when considering a Jaccard distance measure, we should normalize the length of the document.

## 5.3 Clustering setup

Before performing clustering on the data we have to determine which clustering algorithm to use, and which distance measure. This can be achieved by clustering with various setups and evaluating the results.

Before attempting to cluster each possible setup we have to determine whether each distance measure is suitable for the data. The Jaccard distance measure is not suitable for clustering sessions in which multiple similar workflows occur. This is due to the fact that it is not independent of document length. This will have to be determined manually by analyzing some sessions.

To determine the number of clusters to use we plot the WCSS, Silhouette Score, and Calinski-Harabasz Index for 2 to 20 clusters. Each of these three clusters will have an optimal number of clusters. For the Silhouette Score and Calinski-Harabasz Index, this will be the maximum value, while for the WCSS this will be the 'elbow'. We will take a majority vote for the optimal number of clusters. If we have three distinct values we will choose one of the values that maximizes the three values as much as possible. As an example, the Silhouette Score can have multiple high values. If the second-highest value happens to co-exist with the highest value of another plot this can be seen as a good value.

These cluster scores will allow us to determine the optimal clustering setup to use.

## 5.4 Clustering

From the clustering setup, we can determine the optimal clustering algorithm and distance measure. However, the number of clusters could slightly increase. This requires us to create the plots for WCSS, Silhouette Score, and Calinski-Harabasz Index again, however, we can reduce run-time by restricting the range of possible

clusters.

First, we will perform clustering utilizing the vectorization of documents mentioned previously. This allows us to perform clustering on a single client.

Secondly, we will investigate the possibilities of clustering with data from different clients in order to compare them or account for additional workflows. There will be two approaches to consider. The first will remove client-specific data during the vectorization of documents. The second will cluster using the task-based vectors.

## 5.5 Analysis

After clusters have been determined, we will analyze these results.

### 5.5.1 Centroids

To determine an initial meaning for the clusters we set out to analyze the centroid of each cluster. These will be the sessions with the least average distance to other sessions within the cluster. This session will then be manually analyzed.

### 5.5.2 Task division

To provide further insights and automate the analysis of clusters we will determine the tasks being performed in the cluster. Tasks across all sessions in the cluster will be counted. These counts will be sorted, and normalized to a range of [0, 1] to get results as seen in Table 5.1.

| Relevance | Task                       |
|-----------|----------------------------|
| 1.00      | search.configurator        |
| 0.38      | search.search              |
| 0.25      | search.vehicle.details     |
| 0.24      | quotation.quotation.create |

**Table 5.1:** Example task distribution of a cluster

In this example cluster, we can see that these users focus on searching and configuring cars. After viewing the vehicle details a quotation is created. By extracting the most relevant tasks we can get a clearer picture of what is going on in the cluster. An overview of tasks is provided in Appendix A.

### 5.5.3 Role distribution

To further understand what people are performing these tasks for, we will plot the distribution of roles for each of the clusters. This provides further insights into the specific tasks associated with client roles.

### 5.5.4 Page journeys

To investigate the specific order in which requests are made, we will investigate the page journey a user takes through the application during a session. This will be the ordered list of pages visited by the user. Though BetterBe does not track specific attributes to this end, the 'http\_referrer' attribute specifies the URI that the call originates from. Changes in this attribute correspond to a change of page.

#### Conversion from session to journey

First, we consider the set of unique values for the 'http\_referrer' given by  $X$ . As these URIs contain many identifiers, user parameters, and data parameters we create a Regular Expression (REGEX) for each page. This creates a mapping of  $REGEX \rightarrow page\_name$  in which each 'http\_referrer'  $\in X$  has a corresponding REGEX.

We can now create a list of page\_names for a certain session. We will reduce this list by eliminating calls having the same page as the previous call. In example [a, b, b, c, b, a, a] will be reduced to [a, b, c, b, a]. These repeated entries for a page are due to the fact that a page can make multiple API calls to load all the data, or perform another API call due to user activity on the page.

#### Common repetitions

Further analysis into these page journeys can provide insight into common sets of pages visited in a certain order. Such a set of pages [a, b, c] might be repeated several times throughout the session. Locating these repetitions will be done utilizing a rolling hash, discussed in Section 2.4. The minimum number of repetitions within a session and the size of a repetition will be determined. These common repetitions will then provide insights into common actions performed in sessions.

### **5.5.5 Report generation**

Finally, to summarize all the results, and automate the process a report will be generated. This report will collect data on a specific client during a specified time frame. The report will summarize the data, and provide insights into the resulting clusters through a task distribution, role distribution, and a page journey of the centroid. It will also provide common repetitions throughout the sessions.

## **5.6 Usability**

The effort of improving the usability of the application will focus on various aspects. First, an interview will be conducted to have an overview of the current state of affairs. We will then focus on improving the usability through documentation, tailoring pieces of training, and guiding design/ development efforts.

### **5.6.1 Interview**

The interview will focus on the current efforts being put into improving the usability of the application. It should also provide insights into the technical knowledge of people involved with the results of the project to tailor the generated report.

### **5.6.2 Common workflows**

By gaining insights into the common workflows of users, the documentation can be updated to provide concrete examples of how users can perform these workflows. This will improve their understanding of the application, and lower the learning curve by straying away from a trial-and-error approach.

Additionally, by analyzing the workflows of specific clients, their training can be tailored toward these workflows. This improves the ease of learning, and subsequently reduces the required training time for a client.

Finally, these workflows will provide insights into the most-used components of the application. Allocating development efforts according to the importance of tasks can improve usability over time.



### **5.6.3 Common repetitions**

By analyzing the common repetitions, we can identify common actions that are repeated by users often. If we find that there is no efficient way to perform a task, a page can be redesigned, or a new page can be created to support such a task. This will reduce the required time to perform these tasks.



# Data Processing

This chapter details how the data gathered from BetterBe was preprocessed to perform clustering.

## 6.1 Data preprocessing

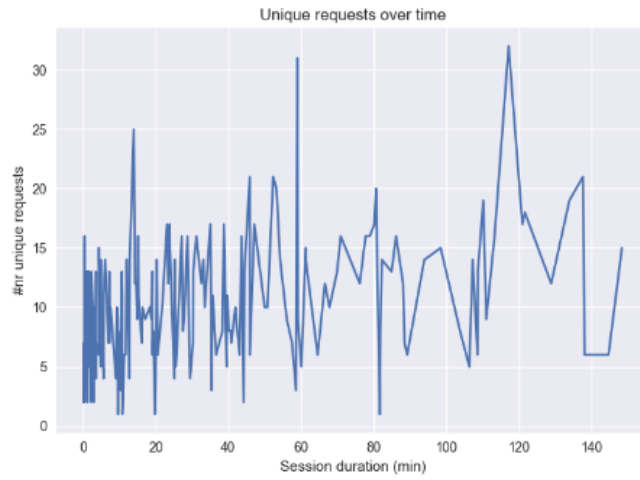
Sessions have been determined by grouping calls by username and session hash. We will now further process this data.

### 6.1.1 Sessions and workflows

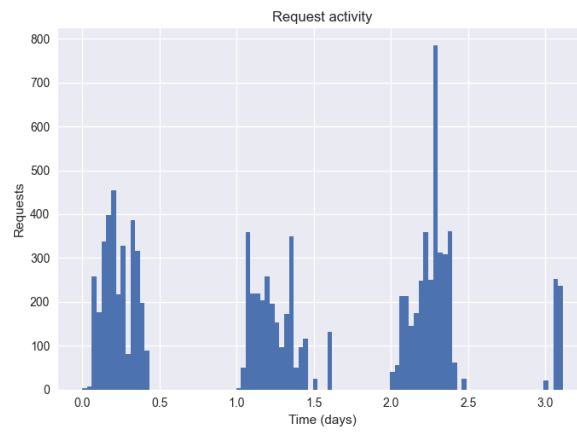
First, we have to account for the fact that a user might perform multiple workflows within a single session. This will be inferred from analyzing the average number of unique requests by session length. These results can be seen in Figure 6.1. Though there is variation in the total number of unique requests this does not seem to significantly change over time. The number of unique requests does not double even though we increase the length of the session several times. Thus, we do not have to further split these determined sessions.

### 6.1.2 Whole sessions

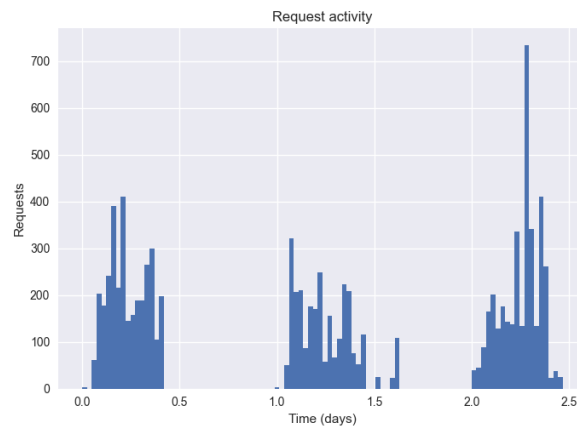
From Figure 6.2 we can observe that there is downtime in the request activity. This makes it easier to split the data in these gaps. This process is automated by searching for this large gap on the last day of data, but manual analysis of new datasets is still required as this gap might not exist for every dataset. Thus, we can split off the 4<sup>th</sup> day of data to avoid incomplete sessions as seen in Figure 6.3.



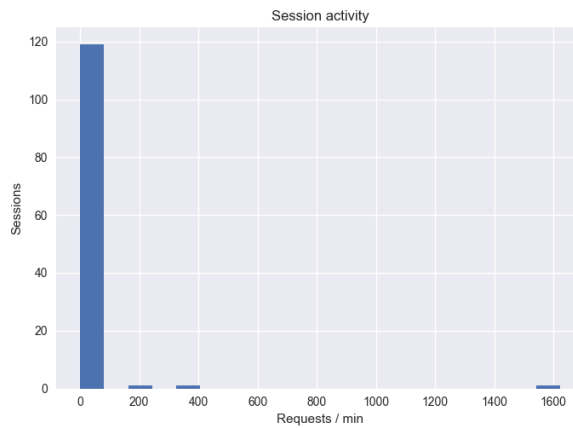
**Figure 6.1:** Number of unique requests for each session length



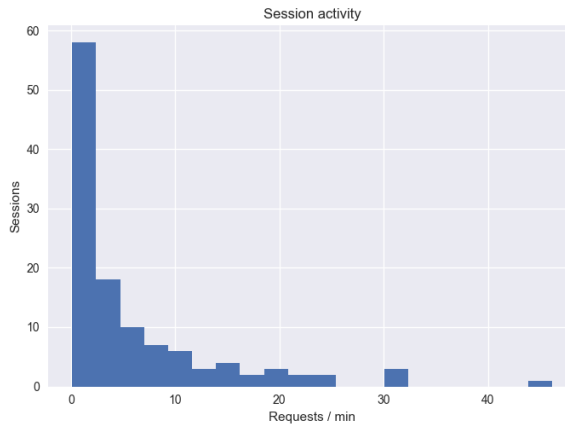
**Figure 6.2:** Request activity over 4 days



**Figure 6.3:** Filtered request activity to remove incomplete session of the final day



**Figure 6.4:** Activity within sessions



**Figure 6.5:** Filtered activity within session to remove outliers

### 6.1.3 Human users

On average users make 24 requests per minute, but from Figure 6.4 we can observe that this is heavily skewed due to a few very active sessions. As we are clustering workflows from users, we will remove these outliers. Once removing outliers above 100 actions per minute, we get a session activity seen in Figure 6.5 where the average becomes 6 requests per minute.

### 6.1.4 Irrelevant calls

As previously discussed in Section 5.1.4, resource calls are removed from the sessions. These resource calls account for 60% of total API calls.

## 6.2 Tokenization of a workflow

After taking the relevant parameters and tokens from the API calls we can further analyze these for the specific domain we are working in. URIs at BetterBe contain specific parameters and identifiers that we will analyze to filter and create additional tokens.

### 6.2.1 URIs

Call type and endpoint can directly be taken as a token. Similarly, user roles is a list of tokens. The API endpoint URI will be tokenized based on delimiters such as "?", "/" and ",". This will split the URI into its layers and parameters. The URIs can contain 6 different parameters.

- *l* (string): Layout identifier. Specific to a dataset.
- *cl* (integer): Calculation layout. Specific to a dataset.
- *ls* (integer): Layout set. Specific to a dataset.
- *r* (integer): Relation identifier (company).
- *p* (integer): Person identifier (from relation).
- *pr* (integer): Profile. Outdated and without significance.

The *l*, *cl*, and *ls* parameters correspond to the layout state of the website. This is specific for each customer, but can provide information on the current actions of the user.

The *r* and *p* parameters correspond to specific companies and their people. Their values are not considered important as they do not provide information to distinguish between different user actions.

The *pr* parameter can be filtered as it is outdated, and has no impact. It is still sometimes passed by clients that have not updated their calls. Thus, the parameters will be translated into the following tokens: ["*l*:value", "*cl*:value", "*ls*:value"].

A request to `"/directlease/nl/management/nl_NL/l:default,ls:1,cl:57/quotation"` will then be split into ["`directlease`", "`nl`", "`management`", "`nl_NL`", "`l:default`", "`ls:1`", "`cl:57`", "`quotation`"].

## 6.2.2 Identifiers

The URIs can contain multiple identifiers as integers, dates, or Universally Unique Identifier (UUID)s. These will be filtered out as they hold no significant information to distinguish between user actions. This can be done using REGEX. As an example, `\w{8}-\w{4}-\w{4}-\w{4}-\w{12}` provides the REGEX to filter out UUIDs. These identifier tokens will be removed.

Additionally, by using domain knowledge of the URIs we can filter words such as "nl" which might occur in every request, or words that occur only once.

## 6.3 Vectorization

After we have a set of workflows  $W$ , we translate these into TF-IDF vectors. This requires us to convert our workflows to a set of words/strings, also known as tokens. This process has been described in Section 5.2. Finally, TF-IDF vectors are created according to the provided formulae. This results in non-negative values in the vectors allowing us to utilize the distance measures.





# Determining Optimal Clustering Setup

We first have to investigate the optimal setup for clustering the data. For this, we will use 10.000 calls as a subset of the data. This will allow us to investigate the preferred distance measure and clustering algorithm.

## 7.1 Distance measure

We will start by comparing a Cosine and Jaccard distance measure.

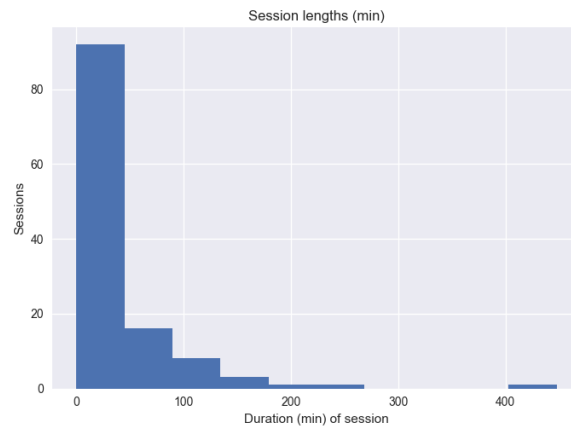
In Figure 7.1, we can observe the distribution of session lengths. By manually evaluating the 3 longest sessions we find that a single workflow of searching and retrieving cars is repeated. This causes the TF-IDF vector of these sessions to be a multiple of similar sessions containing this workflow only once. This makes Jaccard unsuitable due to the large distance between these similar workflows. For this reason, we will work with a Cosine distance measure.

## 7.2 Clustering algorithms

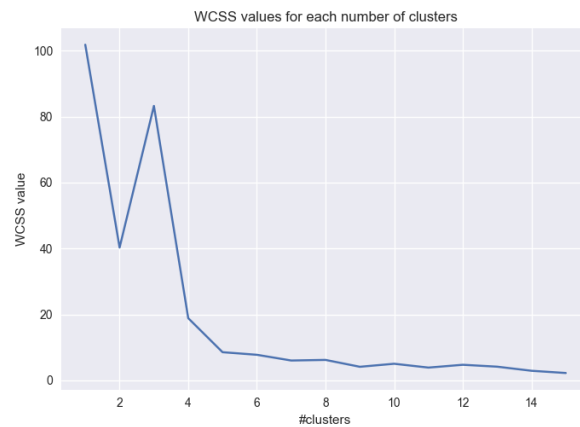
We will now apply both clustering algorithms to determine which one is most suitable.

### 7.2.1 Fuzzy C Means

First, we will cluster using the Fuzzy C Means algorithm combined with a Cosine distance metric.



**Figure 7.1: Session lengths**



**Figure 7.2: WCSS values for each number of clusters**

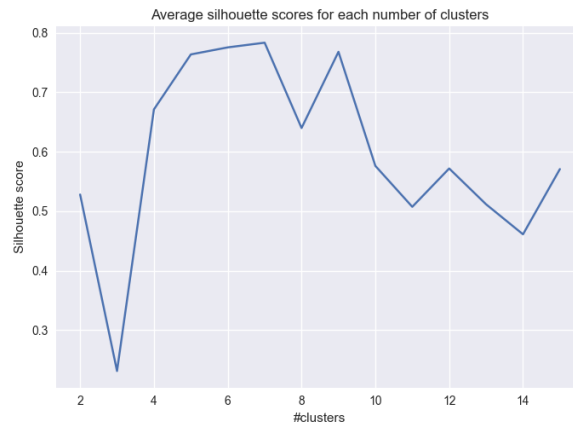
### Determining the number of clusters

First we determine the number of clusters. In Figures 7.2, 7.3, 7.4 we can see the plots corresponding to WCSS, Silhouette and Calinski Harabasz.

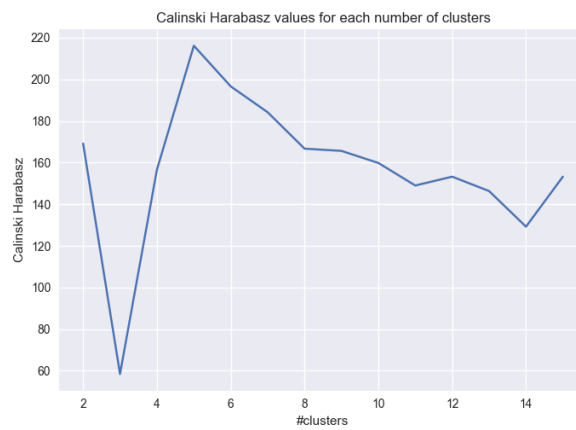
In the WCSS plot we can observe that the elbow occurs at 5 clusters. The Calinski Harabasz provides the maximum score at 5 clusters, while the Silhouette has one of the highest scores at 5 clusters. Thus, we will cluster using 5 clusters.

### Results

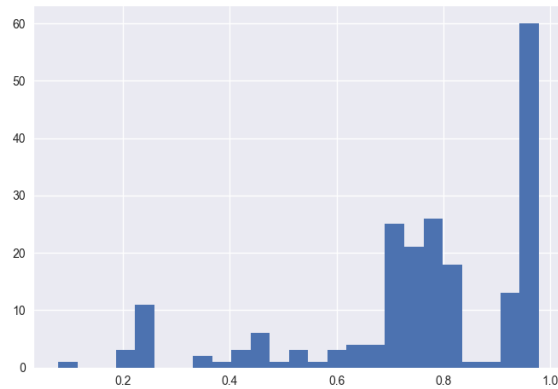
We have 5 clusters of which the distribution can be seen in Figure 7.6. While overlapping clusters are allowed during clustering, we will contribute each session to the closest cluster in the results. To further evaluate the clustering, we can observe the



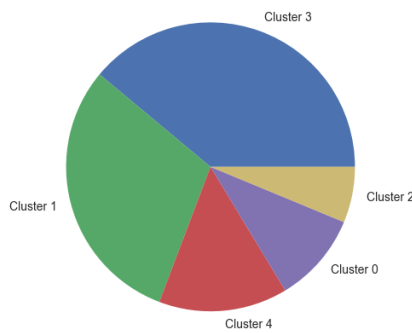
**Figure 7.3:** Silhouette score for each number of clusters



**Figure 7.4:** Calinski Harabasz score for each number of clusters



**Figure 7.5:** Distribution of Silhouette values for each session in the clustering



**Figure 7.6:** Cluster sizes

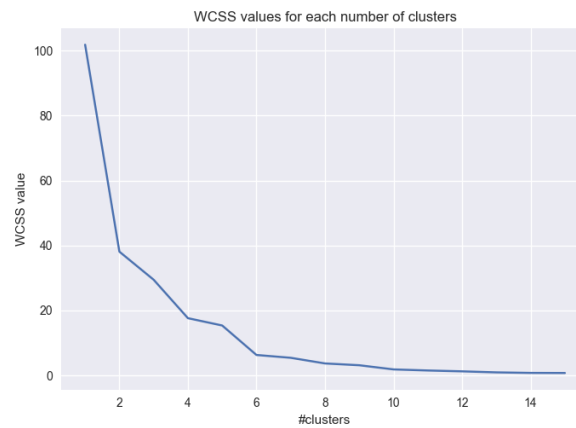
Silhouette values for each session in Figure 7.5. As we have no negative values, we can conclude that their clusters have no significant outliers. Additionally, in Table 7.1 we can see the average Silhouette scores for each cluster. The overall average Silhouette score is 0.76.

The majority of sessions fit well into their clusters with most having a value over 0.6. None of the sessions have a negative value. Clusters 1 and 3 contain nearly 75% of requests with average Silhouette scores of 0.96 and 0.70.

We should note that the Fuzzy C Means algorithm is not deterministic. Thus, we ran the algorithm 5 times. In each of these runs, 5 clusters was the optimal value with similar results.

| Cluster | Size | Average Silhouette Value |
|---------|------|--------------------------|
| 0       | 21   | 0.62                     |
| 1       | 63   | 0.96                     |
| 2       | 13   | 0.93                     |
| 3       | 81   | 0.70                     |
| 4       | 30   | 0.55                     |

**Table 7.1:** Silhouette values for each cluster



**Figure 7.7:** WCSS values for each number of clusters

## 7.2.2 Agglomerative

Secondly, we will cluster using the Agglomerative clustering algorithm combined with a Cosine distance measure.

### Determining the number of clusters

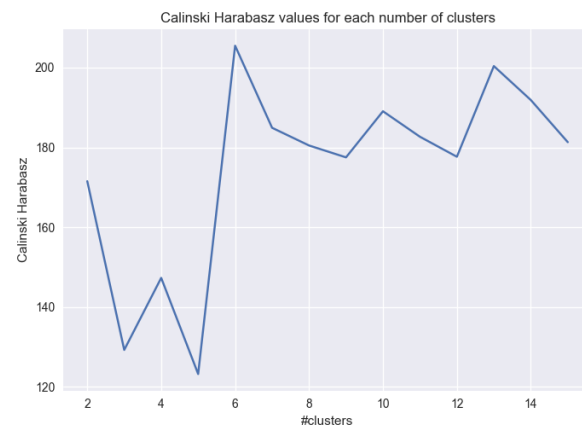
First, we determine the number of clusters. In Figures 7.7, 7.8, 7.9 we can see the plots corresponding to WCSS, Silhouette and Calinski Harabasz. In the WCSS plots, we can observe that the elbow occurs at 6 clusters. The Silhouette and Calinski-Harabasz plots both have peaks at 6 clusters as well. Thus, we choose to plot 6 clusters.

### Results

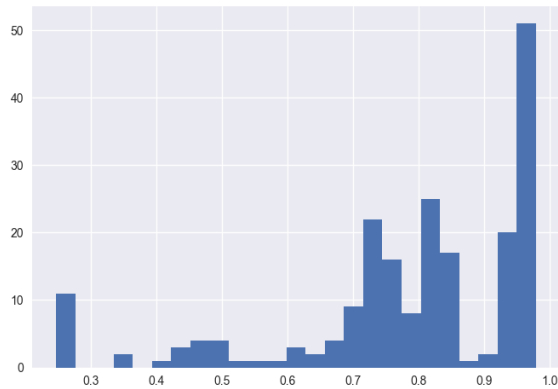
We have 6 clusters of which the distribution can be seen in Figure 7.11. To further evaluate the clustering, we can observe the Silhouette score for each session. This can be seen in Figure 7.10. Thus, it seems that the majority of sessions fits well into it's cluster.



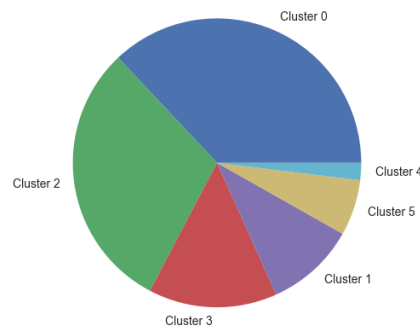
**Figure 7.8:** Silhouette score for each number of clusters



**Figure 7.9:** Calinski Harabasz score for each number of clusters



**Figure 7.10:** Silhouette values for each session in the clustering



**Figure 7.11:** Cluster sizes

To further evaluate each specific cluster, we can provide the average silhouette values for each cluster. This can be seen in Table 7.2. We can observe two large clusters with a value of 0.77 and 0.96. The average Silhouette score is 0.78.

### 7.3 Optimal setup

First, we determined that the Cosine distance was optimal for the dataset. Secondly, we analyzed the different clustering algorithms. Both were clustering in a similar manner which becomes apparent from their identical clusters. Clusters 1 & 0, 2 & 5 and 4 & 3 both have the same size and Silhouette value. However, Agglomerative was able to reduce the number of outliers by creating an additional 6'th

| Cluster | Size | Average Silhouette Value |
|---------|------|--------------------------|
| 0       | 77   | 0.77                     |
| 1       | 21   | 0.62                     |
| 2       | 63   | 0.96                     |
| 3       | 30   | 0.55                     |
| 4       | 4    | 0.74                     |
| 5       | 13   | 0.93                     |

**Table 7.2:** Silhouette values for each cluster

cluster to slightly raise the Silhouette score.

Though there doesn't seem to be a large difference, we will opt to use the Agglomerative clustering. It performed slightly better by finding a 6'th cluster, but also provides a deterministic approach compared to Fuzzy C Means clustering.



# Clustering

In the previous chapter, we investigated the optimal clustering setup using a subset of the data. We will now perform the clustering on 1271 sessions spanning 2 months of data to further analyze the sessions in depth. We will use the Agglomerative clustering algorithm utilizing a cosine distance as previously discussed. The number of clusters will be used as an indication and can serve to limit the search range. This can be especially useful when considering even more sessions as the Silhouette Score has a complexity of  $O(N^3)$ , where  $N$  is the number of sessions.

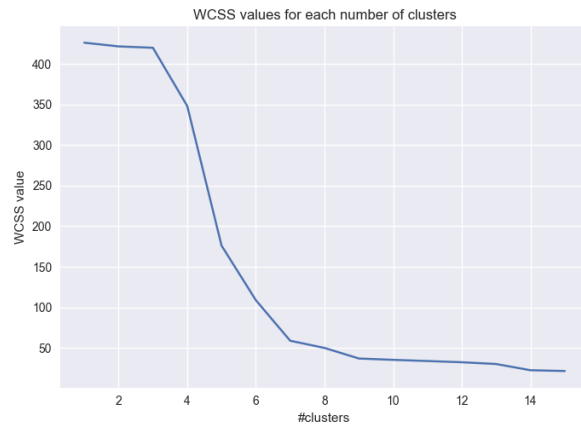
## 8.1 Clustering

As we are taking a longer span of data, we will again determine the number of clusters as less frequent clusters can arise in the larger data.

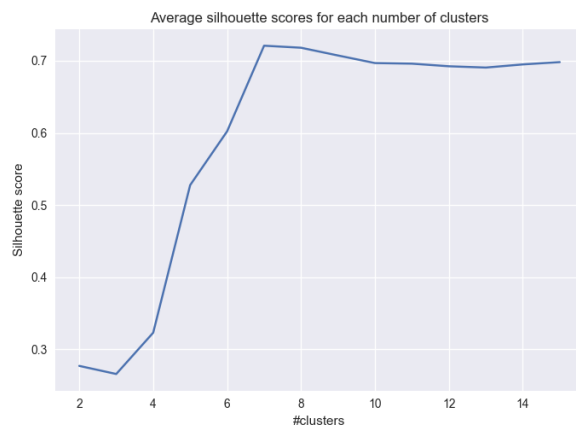
In Figures 8.1, 8.2, 8.3 we can observe that 7 clusters are needed to optimally cluster the data.

## 8.2 Evaluation

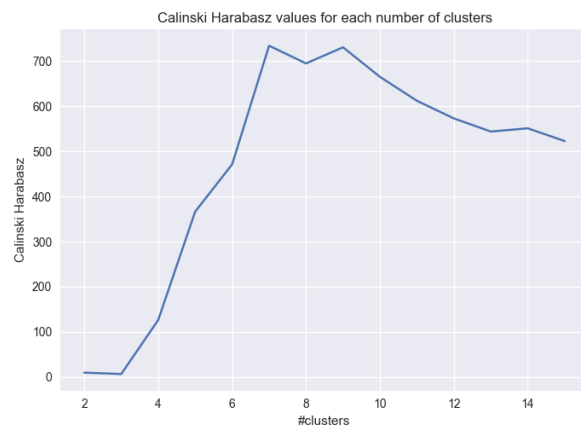
We can observe a similar distribution in clusters to the subset of data, however, a smaller 7<sup>th</sup> cluster has been created. With an average Silhouette value of 0.72, we have a few more outliers. We will start by investigating the meaning of each cluster to provide a human understanding. We will do this by analyzing the centroids, tasks, and role distribution.



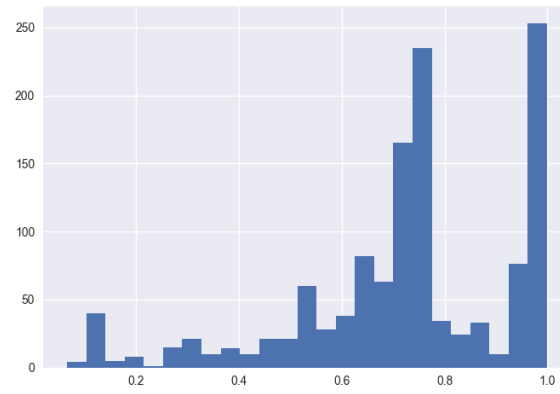
**Figure 8.1:** WCSS values for each number of clusters



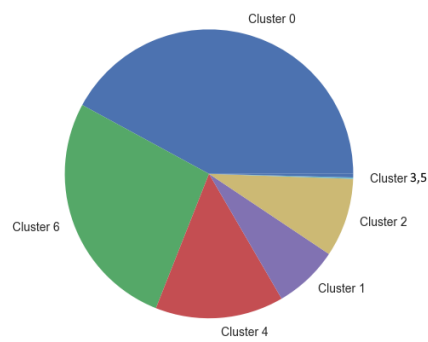
**Figure 8.2:** Silhouette score for each number of clusters



**Figure 8.3:** Calinski Harabasz score for each number of clusters



**Figure 8.4:** Silhouette values for each session in the clustering



**Figure 8.5:** Cluster sizes

## 8.3 Cluster meanings

### 8.3.1 Centroids

Initially to determine the meaning of clusters we set out to analyze the centroids of each cluster. These will be the sessions with the least distance to the other sessions within the cluster. This can be used as a representative of the cluster and will be manually analyzed.

- Cluster 0: Sales person searching/configuring cars and working with quotations.
- Cluster 1: Administrator/dladmin person mainly working on the LOL-code and table data.
- Cluster 2: Data/sales/voertuig data person performing searches and subsequently working in the table data.
- Cluster 3: Data view/sales person searching and viewing a table to download the contents.
- Cluster 4: Sales support person searching/configuring cars and working with quotations.
- Cluster 5: Admin person looking into policies/requests/transactions before creating a report.
- Cluster 6: Sales person searching/configuring cars and working with quotations.

By looking at these centroids we can certainly get an initial impression of the clusters, but for clusters in similar areas such as clusters 0, 4, and 6 it requires more manual work to find what differentiates these clusters. To gain deeper insights into the clusters we attempt to identify what tasks are being performed within these clusters.

### 8.3.2 Task division

In Section 4.2.1, we already discussed roles, tasks, and permissions. In Section 5.2.3 it was discussed how these tasks could be determined. We will now discuss the meaning of the clusters using the tasks. The task distributions per cluster can be found in Appendix B which we manually analyzed.

- Cluster 0: Searching and working with quotations. More focus on listing quotations than creating them.
- Cluster 1: Viewing calculation data and performing searches. Also performing calculation edits.
- Cluster 2: Searching and viewing data in the calculations.
- Cluster 3: Performing searches, viewing developer request info/API calls, and managing quotations relations/people.
- Cluster 4: Searching and working with quotations. Not creating new quotations but rather viewing quotations and looking into calculations data.
- Cluster 5: Viewing calculation tables, performing searches, and managing quotations relations/people.
- Cluster 6: Searching and working with quotations. More focus on creating quotations.

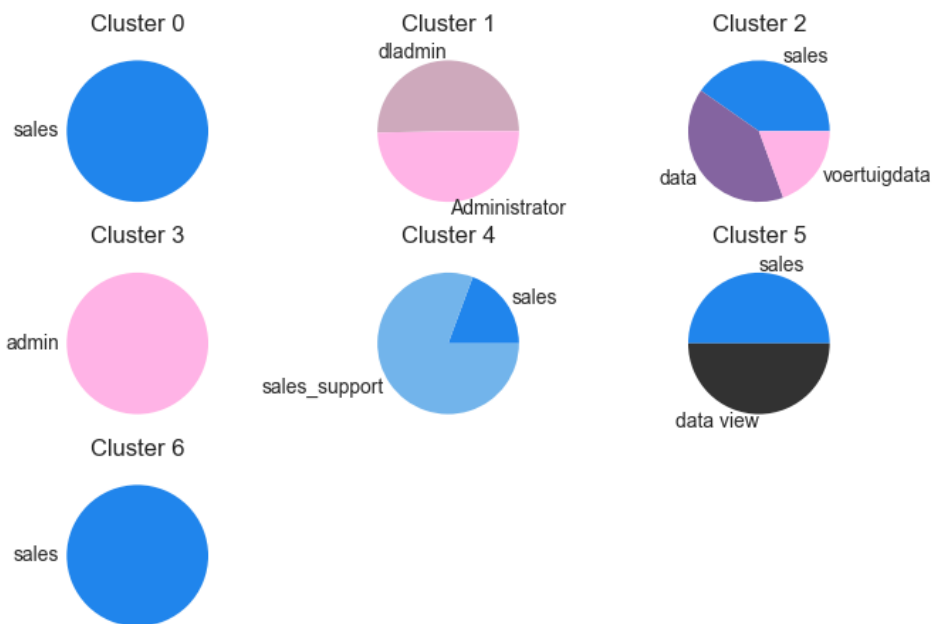
Thus, we see that we can further differentiate between the clusters. As an example, cluster 4 works with searches and quotations but doesn't create quotations. Rather it seems to verify them by looking into calculation data and vehicle details.

### 8.3.3 Role distribution

In Figure 8.6, we can observe the role distribution between the clusters. Clusters 0, 4 and 6 are mainly dominated by the *Sales* and *Sales\_support* roles. Taking into account the previous results we can conclude that these roles are mainly concerned with quotations and that *Sales\_support* indeed has a supporting role in that they don't create quotations, but rather check them.

Clusters 1 and 3 contain 3 different types of admin. *Dladmin* and *Administrator* are custom roles from the company while *Admin* is a global role from BetterBe. Administrator roles are more concerned with managing data, calculations, and requests. These management tasks include deleting objects.

Finally, clusters 2 and 5 are concerned with viewing data and calculation tables. These also contain the *Data view*, *Data* and *Voertuigdata* roles. However, it might seem strange to also find *Sales* here. However, this is explained by the fact that any user with a *Data* role also has a *Sales* role. Furthermore, a portion also has the *Voertuigdata* role.



**Figure 8.6:** Role distribution

# Client comparison obstacles

Previously in Chapters 7 and 8 we have seen how we can gain insights into how LSM is being used. However, this was limited to a single client. To gain a better overall understanding it is vital to not only focus on a single client as they have a wide range of goals which causes them to use the application differently. As an example, Directlease focuses on using the application as a sales tool, while Arval uses it to display car data. Other clients such as Atlon focus more on the calculations. While they all use the system to manage their car data, their goals differ significantly. Currently, gathering this data requires the generation of multiple reports.

## 9.1 Limitations

Currently, multiple attributes are being utilized in the clustering that are client-specific. These include roles and layout parameters. Different clients have various amounts of roles with different sets of tasks making it impossible to create a nice mapping between them. Layout parameters are integers, however, these can have different meanings for different clients.

We will provide two approaches for dealing with these limitations. The first approach is to simply remove these parameters. The second approach will translate sessions into task vectors as seen in Section 5.2.3, and attempt to cluster on this. In both cases, we can compare the results with the previous findings in Chapter 8.

## 9.2 Removal of parameters

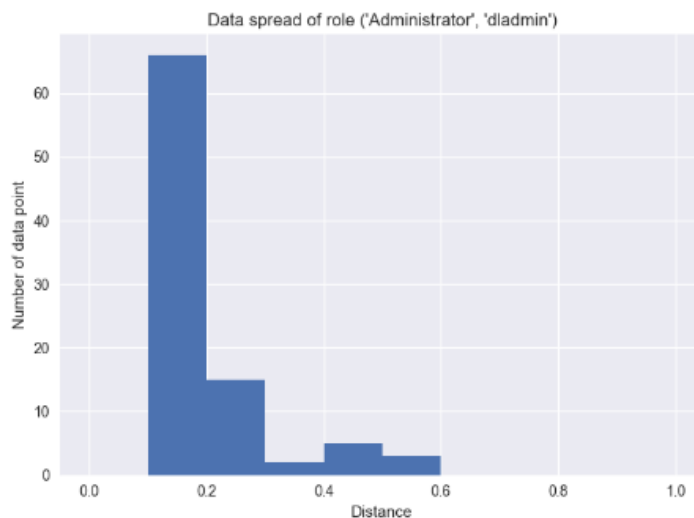
First, we will look into clustering without the client-specific parameters, and compare this to the original clustering. The results can be found in Table 9.1

| #vectors | Outlier removal | #clusters | Silhouette score |
|----------|-----------------|-----------|------------------|
| 1311     | YES             | 20        | 0.53             |
| 1311     | NO              | 21        | 0.38             |
| 155      | YES             | 9         | 0.70             |
| 155      | NO              | 22        | 0.50             |

**Table 9.1:** Clustering results from removing parameters

When clustering 155 sessions, the number of clusters increases from 6 to 22 when compared to the original approach including client-specific data. However, 7 of these clusters have a size of 1 with a total of 12 clusters having a size of 5 or smaller. 4 of these clusters of size 1 correspond to admin roles while there are only 13 data points associated with admin roles. To explain this we can look at the distances between data points within the admin role.

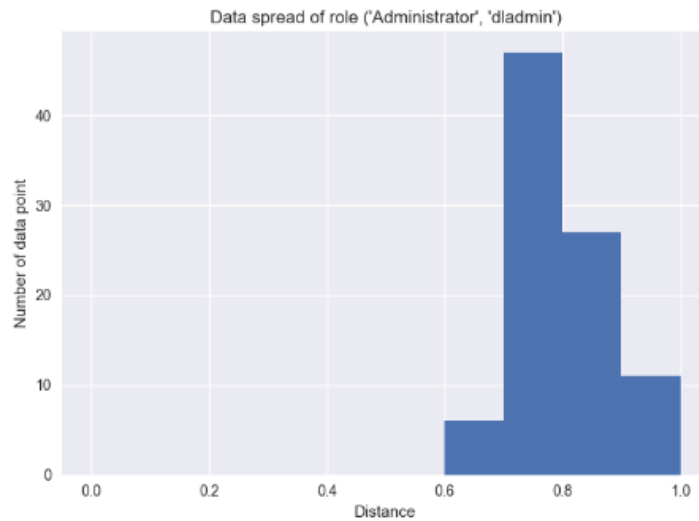
To calculate the spread within the admin role we calculate the average distance to other admin data points for each data point. In Figure 9.1, we can observe the spread of these average distances before removing client specific data. If we remove the role and layout parameters, we get the spread seen in Figure 9.2. From this we can conclude that there is a high degree of spread in activity within the admin role which makes sense as the can perform a wide range of activities.



**Figure 9.1:** Distance spread within admin role including client specific data

In an attempt to reduce the number of clusters created we can filter the outliers from the data. As an example, we can remove sessions with a minimum distance of





**Figure 9.2:** Distance spread within admin role excluding client specific data

0.2 to the nearest other session. This is where a significant drop-off occurs in the number of sessions having a higher value. For the other roles, almost no sessions belong in this range.

For 155 sessions this reduces the number of clusters to 9 and increases the Silhouette score from 0.50 to 0.70. However, as we increase the data size, and with it the number of data points, the chance that an outlier has a distance larger than 0.2 to the nearest neighbor decreases. This results in clusters of size 2 and larger. This becomes clear when we perform clustering on 1311 vectors. After removing the outliers the number of clusters only decreases by 1 from 21 to 20. Furthermore, such filtering can filter large portions of admin data due to its small portion of data and high distances. Thus, a better method of filtering outliers would be required.

### 9.3 Task based approach

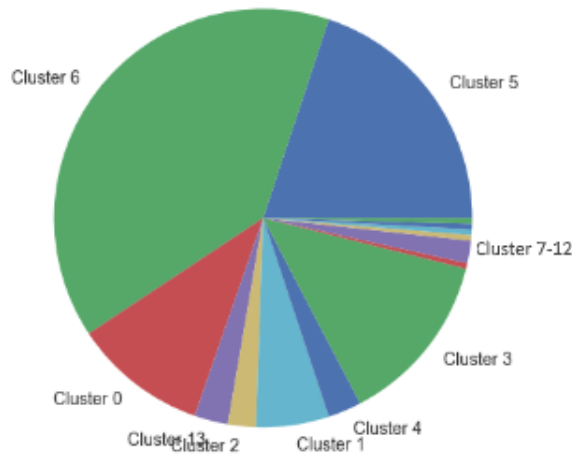
In Section 5.2.3 we have already seen how to determine which tasks are being performed within a session, and how we can translate these into a vector. This allows us to take a similar clustering approach by replacing the TF-IDF vector with the task-based vector.

The results of the clustering using these vectors can be seen in Table 9.2. We now have an increase from 6 to 14 clusters when considering 155 vectors. However, as we can see from Figures 9.3 and 9.4 this is again due to a large amount of small clusters. If we filter the outliers we are reduced to 7 clusters as seen in Figure 9.5.

| #vectors | Outlier removal | #clusters | Silhouette score |
|----------|-----------------|-----------|------------------|
| 1311     | YES             | 12        | 0.57             |
| 1311     | NO              | 16        | 0.37             |
| 155      | YES             | 7         | 0.62             |
| 155      | NO              | 14        | 0.56             |

**Table 9.2:** Clustering results from task based approach

Removing the outliers with larger data again becomes troublesome in this manner with 1311 vectors resulting in 12 clusters. This becomes clear from 1311 vectors as seen in Figure 9.6 where over half of the clusters belong to admin roles.



**Figure 9.3:** Cluster sizes task based approach

## 9.4 Outlier removal

As a possible solution, we can ignore clusters under a certain threshold size rather than finding outlier data points. As these small clusters only consist of a small portion of data this should not be a problem when analyzing common use cases of the application.

## 9.5 Summary

We investigated two different approaches to clustering sessions without client-specific data.

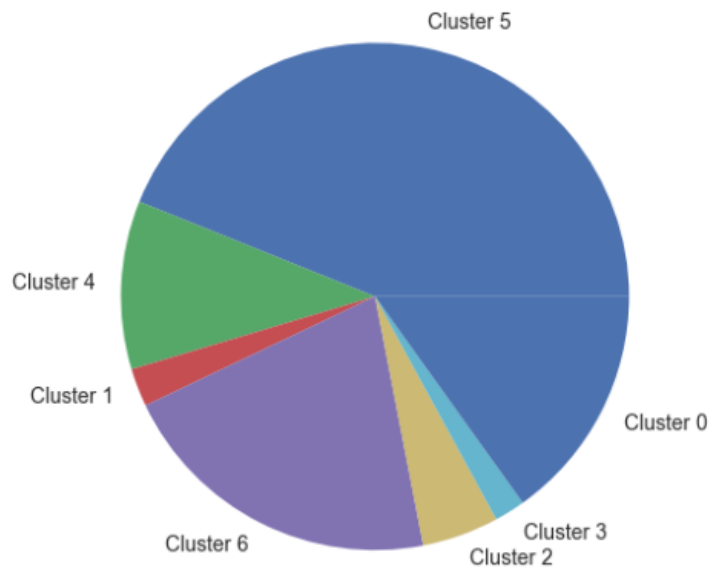


**Figure 9.4:** Role distribution within clusters

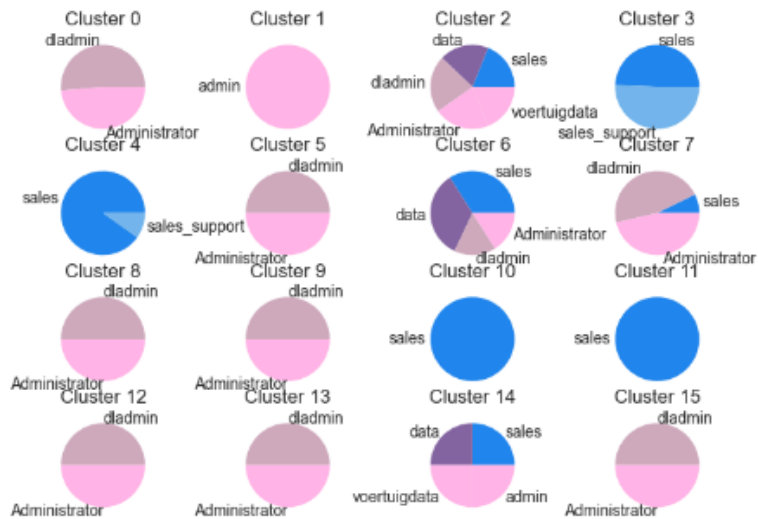
By removing the parameters the number of clusters increases significantly. We observed that this is due to the large variation in tasks performed by certain roles. This creates many outliers in the data resulting in small clusters. As the data size increases these become difficult to filter out.

Secondly, we looked into clustering based on tasks. This can slightly outperform the first approach in terms of the Silhouette score, however, it still suffers from the same issue of small clusters.

A possible solution to this problem is to filter out data corresponding to the clusters under a certain threshold size. These will be groups of outlier data points that can be filtered out.



**Figure 9.5:** Cluster sizes task based approach filtering outliers



**Figure 9.6:** Role distribution within clusters 1311 vectors

# Page Journeys

In previous chapters, we always abstracted from the order in which calls were made, and thus did not take the temporal relation between calls into account. By abstracting from these temporal relations we could not investigate the 'journey' a user took through the application in their session. In this chapter, we will investigate the order in which calls are made to determine the path that was taken throughout the application. We can then locate repetitions in user behavior in sessions as discussed in [27].

## 10.1 Determining page journeys

In order to convert the sessions to page journeys, we first gathered all possible 'http\_referrer' entries, which are URIs, over 10.000 calls from different clients in a set  $X$ . We then manually created REGEXs until we covered the entire set  $X$ . This was needed as the URIs contained various parameters, optional fields, and identifiers. This created a list of 51 pages. We should note that 7 pages already account for 95% of data, and 13 pages account for more than 99% of the data. This highlights the fact that pages relating to accounts, labels, request viewing, preferences, etc. are rarely used. This is to be expected as data related to accounts, labels, and preferences rarely require editing, and pages such as request viewing are only required for debugging purposes. The most used pages relate to searching, configuring, quotations, tables, and LOL-code.

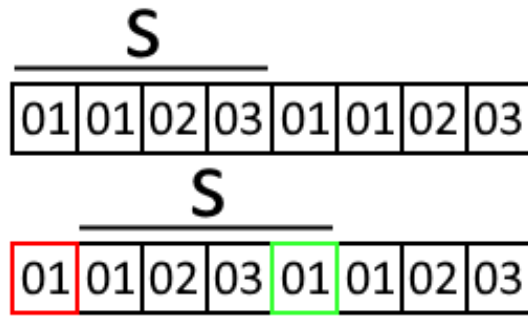


Figure 10.1: Rolling hash sliding window approach

## 10.2 Rolling hash

### 10.2.1 Representation

We can now convert our sessions to a list of pages, which we will reduce, and convert to an integer array by creating an identifier for each page in the range of  $[0, 50]$ . This integer array can then be represented as a polynomial as discussed in Section 2.4.1. This representation is preferred as it allows us to shift the sliding window in  $O(1)$  compared to  $O(n)$  if we were to represent it as an array.

### 10.2.2 Repetitions

To locate repetitions of a fragment  $K$  representing a list of pages, we iterate over possible sizes of our fragment  $K$ . These fragments will have a minimum size of  $s$ , and have to occur at least  $d$  times.

For each size  $s$  we create an initial rolling hash corresponding to the first  $s$  entries in the array. This corresponds to the top array in Figure 10.1. In order to continue with the next set of  $s$  pages, we remove the left-most page in the hash and add the next page according to the operations discussed in Section 2.4.3. This is detailed at the bottom of Figure 10.1 and can be executed in  $O(1)$ .

We store each hash that we encounter and track the first occurrence as  $(index_{start}, index_{end})$ . Once we reach the end of the array we can determine whether a hash occurred more than  $d$  times. We can then use the  $(index_{start}, index_{end})$  to retrieve the pages. This results in a complexity of  $O(n)$  for a single size  $s$ , where  $n$  is the number of pages visited in a session. The maximum number of pages visited in a session is 121.

We search for a minimum of 2 pages as only a single page would provide no

insights. We found that for  $s > 10$ , no repetitions could be found. Thus, we focus on  $2 \leq s \leq 10$  and perform the rolling hash 9 times.

### 10.2.3 Filtering results

By analyzing the results we found that larger repetitions were often composed of a smaller repetition repeated several times. As an example, [03, 01, 02, 03, 01, 02, 03] consists of [01, 02, 03] repeated twice with a fragment at the start. If a repetition matches the pattern of  $[a_1 \dots a_n, (a_1, a_2, \dots a_n)^+, a_1 \dots a_k]$ , where  $[a_1 \dots a_n]$  is another repetition, and + indicates one or more repetitions, we filter it out of the results.

Additionally, a repetition might be a rotation of another. As an example [01, 02, 03] and [02, 03, 01] are rotations. In this case, we remove one of the rotations. While this might remove some relevant data, it proved useful when providing results to BetterBe as it gave a better overview of key repetitions.

## 10.3 Results

After filtering the results, we manually reviewed the repetitions with BetterBe. There were repetitions that they expected to see as they follow the general use cases of the application. However, there were also inefficient workflows and uncommon features. We will provide a summary of these relevant findings that were discussed below.

### 10.3.1 Configure, Search, Quotation

There are multiple repetitions involving Configure, Search, and Quotation which was expected to occur. The inclusion of Quotation is somewhat specific to Directlease as not all companies use the application to create quotations. These pages are often used together as a configure and search is used to find a car after which a quotation can be created.

### 10.3.2 Person, Quotation

This is another set specific to Directlease. Most companies do not store people in the system. It makes sense to see this occur as Directlease does in fact make use of this fact by checking quotations created by specific people.

### **10.3.3 Lol, Table**

This indicates that a developer is working on LOL-code, and switches over the Table page. This is seen multiple times and is often used to check the contents of tables that the user is working with.

### **10.3.4 Vehicle, Table**

This was interesting as it wasn't directly clear how a user could make this leap. It turned out that a feature had once been built to check vehicle data in the tables directly, but this was somewhat of a hidden feature. Determining how this was done wasn't something a user could easily figure out.



# Report Generation

In previous chapters, we have gathered and investigated various data relating to user clustering, role distributions, task distributions, and page journeys. Initially, this was performed on offline data corresponding to a single company to reproduce data, and create consistent figures for the report.

In the future, it can be expected that BetterBe would like to generate this data for other companies with different, newer time frames. To reduce the amount of effort required for this, the process will be automated to easily provide such an analysis in a matter of minutes. The final result will be a generated report. The target audience of this report will be employees of BetterBe.

## 11.1 Generation

### 11.1.1 Steps

The report generation consists of the following steps:

1. **Input:** Prompt user for start date, end date and company.
2. **Data:** Automatically query and retrieve data from Elasticsearch.
3. **Preprocessing:** Preprocess the data for clustering.
4. **Initial clustering:** Perform clustering with a different number of clusters.
5. **Input:** Prompt user for number of clusters using WCSS, Silhouette, and Calinski-Harabasz figures. An explanation is provided to ensure non-technical users can also complete the generation.

6. **Clustering**: Cluster the data.
7. **Analysis - General**: Perform general data analysis relating to the role distribution, user requests, request activity, session lengths, and session activity.
8. **Analysis - Clusters**: Perform analysis on each cluster relating to the role distribution, task division, and page journey.
9. **Generate report**: Generate and download the report for the user.

### 11.1.2 Invocation

The generation can be executed from the command line using the following syntax:  
`main.py [start_date] [end_date] [company_name] [flags]`

The dates are in Elasticsearch syntax where we can use "now-365d" and "now" to include an entire year. The number of requests are capped at 30.000 by default unless overwritten in the flags to avoid long runtimes. We provide the following flags to customize the generations:

- **-al**: Algorithm. The clustering algorithm to use. "fuzzy\_c\_means" or "agglomerative". Defaults to "agglomerative".
- **-ap**: Appendix. Boolean to include WCSS, Silhouette, and Calinski-Harabasz plots in the appendix that were used to determine the number of clusters. Defaults to False, and will not be provided if plots are skipped due to setting the number of clusters directly using the 'cl' flag.
- **-ca**: Caching. If set to True the data retrieved from Elasticsearch is cached. Defaults to false.
- **-cl**: Clustering. Directly specify the number of clusters to use in order to skip the process of generating WCSS, Silhouette, and Calinski-Harabasz plots to determine the number of clusters.
- **-cs**: Cluster size. Can specify a minimum cluster size in the resulting clusters. Defaults to 1.
- **-dm**: Distance measure. Specifies the distance measure to use in the clustering algorithm. "cosine" or "jaccard". Defaults to "cosine".
- **-mca**: Max calls. Species the maximum number of calls to request. Defaults to 30.000.

- **-mcl**: Max clusters. Specifies the maximum clusters for which we determine values in the WCSS, Silhouette, and Calinski Harabasz plots. Defaults to 20.
- **-na**: Name. Name of the report. Defaults to "report".
- **-sa**: Session activity. Specifies a maximum activity allowed within a session. This activity is given by ( $\#nr\_requests / \#minutes$ ) for each session. Plots to determine a value can be found in the generated report.
- **-sl**: Session length. Specifies a maximum session length allowed. A distribution of session lengths can be found in the generated report.

## 11.2 Report

Reports were generated for the 10 largest clients of BetterBe with respect to the number of calls they made. A sample report for "directlease\_nl" can be found in Appendix D.

The data summary describes the number of days, users, calls, and sessions that are being analyzed. This is followed 4 figures:

- User requests: Histogram describing how many calls users make.
- Request activity: Plot describing how many requests are made over time.
- Session lengths: Distribution of session lengths.
- Session activity: Histogram describing the activity within a session described by  $\frac{requests}{minutes}$ .

This is followed by the Workflow section containing the clustering. First, it describes the size distribution of each cluster, and the role distribution for each cluster. For each cluster, it then provides a summary, a role distribution, a task distribution, and a page journey for the session with the least distance to the others in the cluster.

The task distribution might be less understandable for people not involved with the application, but BetterBe employees can interpret these fairly well to understand what is happening in the session. The page journey of a session highlights the order of pages visited in the session. This is done for the centroid of each cluster.

Finally, there is a Repetition section containing information relating to common page journey repetitions found in the sessions.



# Usability

We had an interview with BetterBe to evaluate the current state of affairs related to what is being done to improve usability. This can be found in Appendix C. We will detail how usability can be further increased through documentation, training, and design in the future. These will benefit ease of learning, reduction of errors, and efficiency of the interface design discussed in Chapter 3.

## 12.1 Documentation

In the current state, documentation is only provided in a limited quantity. This makes it difficult for programmers to automate tasks and interact with the application. This usually results in a trial-and-error approach. This difficulty of mapping desired scenarios to actions in the application was described as a lack of understanding in the upward direction [20], and decreases the usability of the application.

Solutions to these issues are providing code snippets for common workflows to describe how certain actions are performed. The generated reports for clients can provide insights into these common workflows of users, and can thus be used to write documentation tailored towards these actions as new users are likely to fall within a cluster of previous users.

It should be noted that many of these actions are currently being performed by users, but this does not mean that they are performed in the best way. There are many 'hidden' features within the application that can help speed up the process. Thus, this documentation is not only for new users but also for existing users. By providing an overview of available features with examples efficiency can increase, and errors will reduce.

## 12.2 Training

Training at companies is currently a way in which BetterBe details global possibilities within the system. This can guide their design process when adopting the software. Thus, their current goal with the training is to provide a high-level explanation of their possibilities within the application before using the application.

However, we know that the majority of new clients will be branches in different countries from existing clients. As an example, companies might start with a single country adopting the software, and slowly roll this out to additional countries. These clients can be expected to perform similarly to their other branches. This allows us to tailor the process by generating a report on their other branch. By tailoring training towards a specific client, and adding emphasis on likely-to-use features within the software the ease of learning will improve.

## 12.3 Design

By analyzing common workflows, and page repetitions, we were able to gain greater insights into how users browse through the application. Two of these repetitions stood out, and we will go into further detail on how we can use these to improve the usability of the application through design changes in the following sections.

### 12.3.1 Vehicle, Table

There are employees at companies that view cars to verify data, and subsequently, make changes in the tables. Finding the data in various tables related to a vehicle can be a long process of viewing multiple tables and editing data. A few years ago a feature was built to simplify this process, but when trying to locate how this worked we could not figure this out within 30 minutes.

As a simpler but less efficient way exists of navigating to these tables manually and making the changes, it cannot be expected that users can find this feature. By including these features in the pieces of training/documentation, and improving visibility, this potentially long manual navigation task can be greatly sped up for a user improving the efficiency of the application.

### **12.3.2 Lol, Table**

It often arises that a developer might have to view a table when editing in LOL-code. Currently, this requires a user to click on the 'calculation tables' tab followed by selecting the correct table. This can be further complicated by tables having version names/numbers. An improvement we discussed was including links within the editor to directly go to the relevant table. This would simplify the process of viewing a table while editing code by shortening the navigation, and eliminating the possibility of navigating to the wrong table.





# Discussion

By investigating user journeys through the application we were able to determine design changes to improve usability. However, these page journeys were not complete. Pages in the application often have 'tabs' as seen in Figure 13.1. These 'tabs' often lead to a new page, but can also change the information displayed on the page without a page change. These changes are not included in the navigation as there is currently no way to track 'tab' changes.

Lastly, we have made multiple recommendations to improve the usability of the application. However, these will take time to implement. Documentation/training will have to be created, and design changes have to be implemented. Thus, evaluating the impact of these changes is currently out of scope for this research.



**Figure 13.1:** Tabs on the quotation page



### **Future Work**

Equipped with the gathered knowledge, BetterBe is able to make design changes, and monitor how user behavior changes over time. However, the data on page journeys used for design changes can be further refined. There is currently no data to track whether a user is currently in tab 'A' or tab 'B' on a page which can cause different pages to be seen as a single page. They can opt to better track user activity by sending additional requests, or embedding requests with an additional parameter to accurately detail where a user currently is. This can provide more accurate user journeys in the future to help further improve usability.

Additionally, BetterBe has expressed an interest in possibly sharing generated reports with clients in the future. A possible use case might be to help them optimize user roles within the application.

Finally, the approach can be simulated at other companies to improve usability. The clustering approach can be replicated, and page journeys can be determined. However, other companies might not have clearly defined tasks for users. Thus, the research can serve to improve usability at other companies with minor adjustments.



# Conclusion

This research aimed to improve the usability of BetterBe's application by analyzing previous usage data of the API. This was split into three research questions.

### **How can we extract workflow patterns from gathered API usage data**

By determining user sessions in the data we were able to extract related user calls. To provide additional insights into these workflow patterns we computed the page journey of a user throughout the application, and which tasks were performed.

### **How can we identify common workflow patterns between users?**

User workflows were converted into vectors to provide a numerical representation of a workflow. In Chapter 8, we have seen that we could cluster the data efficiently after determining that the Cosine distance measure and Agglomerative clustering algorithm yielded the best results in Chapter 7. This allowed us to cluster workflow patterns to identify common workflow patterns between users.

### **How can we improve the usability at BetterBe by analyzing the data and common workflow patterns?**

In Chapter 11, we generated a report to summarize common workflows and provide insights into popular page journeys of users. Through investigating these common usage patterns of clients, and reviewing common repetitions we were able to propose changes relating to documentation, training, and design. These lead to improved ease of learning, efficiency of the design, and a reduction in errors. By targeting these usability factors we have established a clear plan of changes to improve the usability of the application.



# Bibliography

- [1] F. Montesi and J. Weber, “Circuit breakers, discovery, and API gateways in microservices,” *arXiv preprint arXiv:1609.05830*, 2016.
- [2] B. A. Myers and J. Stylos, “Improving API usability,” *Communications of the ACM*, vol. 59, no. 6, pp. 62–69, 2016.
- [3] J. Stylos and B. A. Myers, “The implications of method placement on API learnability,” in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008, pp. 105–112.
- [4] T. Grill, O. Polacek, and M. Tscheligi, “Methods towards API usability: A structural analysis of usability problem categories,” in *International conference on human-centred software engineering*. Springer, 2012, pp. 164–180.
- [5] “Elastic search,” <https://www.elastic.co/>.
- [6] “Kibana,” <https://www.elastic.co/kibana/>.
- [7] J. Karsten, “Identifying user workflows to improve API usability,” 2022.
- [8] M. Halkidi, *Hierarchical Clustering*. Boston, MA: Springer US, 2009, pp. 1291–1294. [Online]. Available: [https://doi.org/10.1007/978-0-387-39940-9\\_604](https://doi.org/10.1007/978-0-387-39940-9_604)
- [9] H. Jiang and S.-J. Lin, “A rolling hash algorithm and the implementation to lz4 data compression,” *IEEE Access*, vol. 8, pp. 35 529–35 534, 2020.
- [10] K. Bhattacharjee, K. Maity, and S. Das, “A search for good pseudo-random number generators: Survey and empirical studies,” *arXiv preprint arXiv:1811.04035*, 2018.
- [11] J. Rasmussen, “Design with the user in mind, and you can’t go awry,” *Comput. Canada*, vol. 22, no. 25, pp. 47–48, 1996.
- [12] J. M. Spool, T. Scanlon, W. Schroeder, C. Snyder, and T. DeAngelo, “Web site usability: A designer’s guide, user interface engineering,” *North Andover MA*, 1997.

- [13] M. F. Zibrán, F. Z. Eishita, and C. K. Roy, "Useful, but usable? factors affecting the usability of APIs," in *2011 18th Working Conference on Reverse Engineering*. IEEE, 2011, pp. 151–155.
- [14] T. R. G. Green and M. Petre, "Usability analysis of visual programming environments: a 'cognitive dimensions' framework," *Journal of Visual Languages & Computing*, vol. 7, no. 2, pp. 131–174, 1996.
- [15] S. Clarke, "Measuring API usability," *Dr. Dobb's Journal Windows*, pp. S6–S9, 2004.
- [16] G. Fischer, "Cognitive view of reuse and redesign," *IEEE Software*, vol. 4, no. 4, p. 60, 1987.
- [17] B. M. Hughes, G. Polson Peter, K. Muneo, and L. Clayton, "Cognitive walk-through for the web," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2002.
- [18] A. Bhattacharjee, "Understanding information systems continuance: An expectation-confirmation model," *MIS quarterly*, pp. 351–370, 2001.
- [19] J. Bloch, "How to design a good API and why it matters," in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, 2006, pp. 506–507.
- [20] M. P. Robillard and R. DeLine, "A field study of API learning obstacles," *Empirical Software Engineering*, vol. 16, no. 6, pp. 703–732, 2011.
- [21] D. Hou and L. Li, "Obstacles in using frameworks and APIs: An exploratory study of programmers' newsgroup discussions," in *2011 IEEE 19th International Conference on Program Comprehension*. IEEE, 2011, pp. 91–100.
- [22] E. Murphy-Hill, C. Sadowski, A. Head, J. Daughtry, A. Macvean, C. Jaspan, and C. Winter, "Discovering API usability problems at scale," in *Proceedings of the 2nd International Workshop on API Usage and Evolution*, 2018, pp. 14–17.
- [23] M. Pearrow, *Web Site Usability Handbook with Cdrom*. Charles River Media, Inc., 2000.
- [24] J. Nielsen, *Usability engineering*. Morgan Kaufmann, 1994.
- [25] M. C. Roy, O. Dewit, and B. A. Aubert, "The impact of interface usability on trust in web retailers," *Internet research*, 2001.



- [26] A. Vecchione, D. Brown, E. Allen, and A. Baschnagel, "Tracking user behavior with google analytics events on an academic library web site," *Journal of Web Librarianship*, vol. 10, no. 3, pp. 161–175, 2016.
- [27] S. Dustdar and R. Gombotz, "Discovering web service workflows using web services interaction mining," *International Journal of Business Process Integration and Management*, vol. 1, no. 4, pp. 256–266, 2006.
- [28] G. Poornalatha and P. S. Raghavendra, "Web user session clustering using modified k-means algorithm," in *International Conference on Advances in Computing and Communications*. Springer, 2011, pp. 243–252.
- [29] J. H. Paik, "A novel TF-IDF weighting scheme for effective ranking," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 343–352.
- [30] Z. Yun-tao, G. Ling, and W. Yong-cheng, "An improved TF-IDF approach for text classification," *Journal of Zhejiang University-Science A*, vol. 6, no. 1, pp. 49–55, 2005.
- [31] S. Qaiser and R. Ali, "Text mining: use of TF-IDF to examine the relevance of words to documents," *International Journal of Computer Applications*, vol. 181, no. 1, pp. 25–29, 2018.



# Tasks

Below we have a list of relevant tasks. It is not a complete list, but rather tasks that we utilized within the sessions gathered from the data.

- Accessory
- Account
  - Account.edit
- Apicall
- Bulkedit
- CalculationChecker
  - CalculationChecker.vehicleids
- CalculationParameterSet
  - CalculationParameterSet.edit
  - CalculationParameterSet.view
- CalculationTable
  - CalculationTable.edit
  - CalculationTable.historic
- Configurator
  - Configurator.vehicleids
- Details
- Label

- List
- Lol
  - Lol.delete
  - Lol.save
- Myaccount
- Optiongraph
  - Optiongraph.view
- Pagelayout
- Pagelayoutset
- Person
  - Person.create
  - Person.delete
  - Person.edit
- Policy
  - Policy.create
- Quotation
  - Quotation.create
  - Quotation.delete
  - Quotation.index
  - Quotation.pdf
  - Quotation.view
- QuotationTemplate
  - QuotationTemplate.edit
  - QuotationTemplate.pdf
- Refdate
  - Refdate.create
  - Refdate.createfull

- Refdate.delete
- Relation
  - Relation.delete
- Report
  - Report.download
  - Report.edit
  - Report.view
- RequestInfo
  - RequestInfo.details
- Resource
- Role
  - Role.edit
- Search
- Selection
- Synclog
  - Synclog.details
- Tracecar
  - Tracecar.vehicleids
- Transaction
- Vehicle
  - Vehicle.copy
  - Vehicle.create
  - Vehicle.delete
  - Vehicle.derive
  - Vehicle.edit



# Cluster Tasks

| Cluster | Relevance | Task                                  |
|---------|-----------|---------------------------------------|
| 0       | 1.00      | search.configurator                   |
|         | 0.48      | quotation.quotation.list              |
|         | 0.38      | search.search                         |
|         | 0.25      | search.vehicle.details                |
|         | 0.24      | search.search.tracecar                |
|         | 0.24      | calculations.lol.view                 |
|         | 0.24      | quotation.quotation.create            |
| 1       | 1.00      | calculations.lol.view                 |
|         | 0.82      | search.search                         |
|         | 0.80      | search.vehicle.details                |
|         | 0.64      | search.search.tracecar                |
|         | 0.53      | search.configurator                   |
|         | 0.28      | calculations.calculationTable.view    |
|         | 0.15      | calculations.calculationTable.content |
|         | 0.14      | quotation.person.manage               |
|         | 0.11      | calculations.lol.edit                 |
| 2       | 1.00      | search.search                         |
|         | 0.69      | search.configurator                   |
|         | 0.42      | search.vehicle.details                |
|         | 0.39      | search.search.tracecar                |
|         | 0.39      | calculations.lol.view                 |
|         | 0.36      | calculations.calculationTable.view    |
|         | 0.24      | calculations.calculationTable.content |

**Table B.1:** Task distributions Clusters 0 - 2

| Cluster | Relevance | Task                               |
|---------|-----------|------------------------------------|
| 3       | 1.00      | search.search                      |
|         | 0.43      | developer.requestInfo              |
|         | 0.39      | search.vehicle.details             |
|         | 0.39      | search.search.tracecar             |
|         | 0.39      | developer.requestInfo.details      |
|         | 0.39      | calculation.lol.view               |
|         | 0.34      | quotation.relation.manage          |
|         | 0.30      | quotation.person.manage            |
|         | 0.17      | search.configurator                |
|         | 0.17      | datamanagement.report.view         |
|         | 0.13      | developer.apicall                  |
| 4       | 1.00      | search.search                      |
|         | 0.92      | search.configurator                |
|         | 0.58      | quotation.quotation.list           |
|         | 0.54      | search.vehicle.details             |
|         | 0.53      | search.search.tracecar             |
|         | 0.53      | calculations.lol.view              |
| 5       | 1.00      | calculations.calculationTable.view |
|         | 0.56      | search.search                      |
|         | 0.22      | search.vehicle.details             |
|         | 0.22      | search.search.tracecar             |
|         | 0.22      | quotation.relation.manage          |
|         | 0.22      | quotation.person.manage            |
|         | 0.22      | calculations.lol.view              |
| 6       | 1.00      | search.configurator                |
|         | 0.32      | quotation.quotation.create         |
|         | 0.21      | search.vehicle.details             |
|         | 0.20      | search.search.tracecar             |
|         | 0.20      | calculations.lol.view              |
|         | 0.18      | search.search                      |
|         | 0.12      | quotation.quotation.list           |
|         | 0.11      | quotation.quotation.manage         |

**Table B.2:** Task Distributions Clusters 3 - 6



# Interview

An interview was conducted with regards to the current state of affairs surrounding usability improvements at BetterBe. We detail the most relevant questions of which the content was used to motivate choices in the report.

## **C.1 Interview**

### **C.1.1 How many clients are currently using the software?**

We have around 10 large clients. There is usually a team of 5 people per client dealing with the software.

### **C.1.2 How will this scale over time?**

The number of clients will likely remain the same, but as most clients are internationally based, the number of countries will increase. It is expected that this could increase the number of teams to 20-30.

### **C.1.3 What is the technical knowledge of users interacting with the software?**

There are three types of users interacting with the software. The first is developers which are mainly involved with coding in LOL and have good technical knowledge. The second is the technical pricers that have less technical knowledge and are involved with prices of policies, tires, etc. These users only interact with a small part of the software. Lastly, there are vehicle releasers that determine which cars are ready for release by checking cars in the system. These have the least technical knowledge.

#### **C.1.4 What is currently being done to improve usability for these users?**

Not a lot. We've had a few talks with clients.

#### **C.1.5 How often do you have this client interaction, and what is the main topic?**

We've only had a single talk with a client with the goal to improve the application, but it was rather limited. Other interactions were based on missing features which occurred only a few times.

#### **C.1.6 How do clients learn to initially use the software?**

We provide them with a few training courses. These are a one-time occurrence at the start of the process of setting up the software for a client.

#### **C.1.7 What is the primary focus of these pieces of training?**

They attempt to show what is possible with the application. This can show them which structures can be supported within their application design. It is still rather global as the application has not been set up for the client at this point.

#### **C.1.8 Do you provide some form of documentation for the clients after they start using the application?**

No, this is fairly limited. The main documents are release notes that contain information on new features.

#### **C.1.9 What would you say are the main limitations of the current approach?**

Users often use features that they manage to find. Once they find a way to tackle a certain problem it might not be the optimal solution. They are not aware of these optimal solutions, and BetterBe is not aware of these inefficient processes. Finding these inefficient processes is not that simple.

### **C.1.10 How far can the generated reports provide useful insights into current application interactions?**

It will show us how people use the application. These points can be taken into consideration for the application as feedback. It can be useful in the development of the application to focus efforts.

### **C.1.11 How well can people at BetterBe translate cluster tasks to workflows in the application?**

Rather well. We can how sessions are distinguished for similar roles. This provides good insights. Additionally, we could find clusters in the Directlease that were expected which verifies the results.

### **C.1.12 What are your thoughts on the current design and layout of the application?**

We are making fewer GUI changes than we would hope. These changes cost a lot of time, but we don't know which are important. Various changes are possible, but it is difficult to estimate whether these changes are worth pursuing.

### **C.1.13 Are you aware of problems that could arise from this design?**

We are aware of some issues that we found ourselves. As an example, it is hard to gather related items. Cars have many attributes, but there are no direct links to the related tables. However, we don't know whether users would want this feature so it is difficult to estimate whether this is worth our time to support.



## Appendix D

# Report generation

In the following pages we can view a sample generated report.

# Directlease\_NI

LeaseServices Management Analysis





# Introduction

## Data Summary

In this report we will provide an analysis of how directlease\_nl utilizes the LSM application. First, we will explore general insights into the data to gain a clear picture of the what is being analyzed. Secondly, we will dive deeper into specific use cases within the application. For these use cases we will provide involved roles, tasks being performed, and a sample workflow through the application for one of these users. Thirdly, we will provide common patterns or repetitions that occur in the sessions. These consist of a set of pages that are often repeated. Finally, the Appendix contains the plots that were used to determine the number of clusters.

**64**  
**Days**

**36**  
**Users**

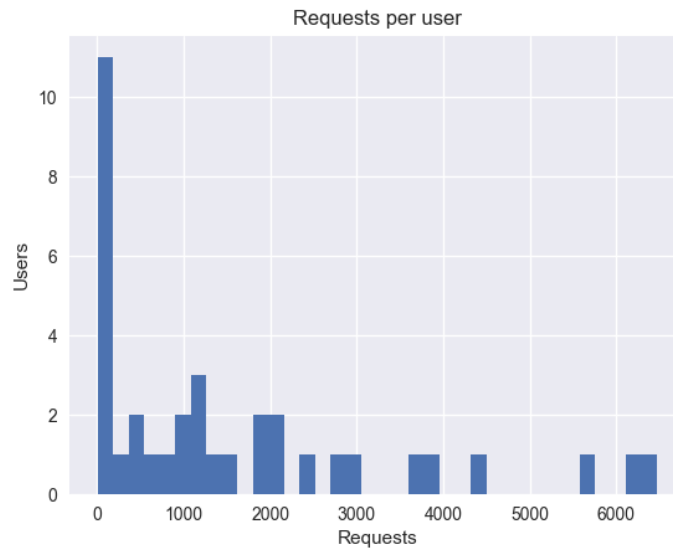
**57318**  
**Calls**

**1685**  
**Sessions**



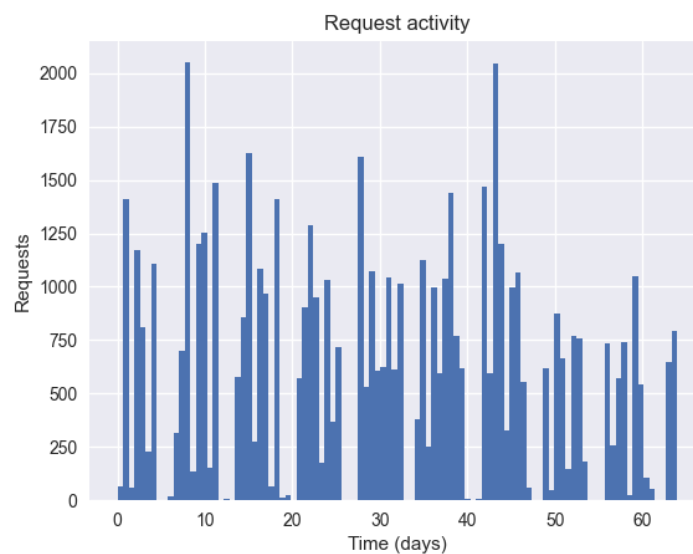
## User requests

The plot below details how many requests users make on average. On the x-axis we have bucketed numbers of requests while the y-axis details how many users in the dataset make this number of requests through the entire time frame.



## Request activity

The plot below details the activity of requests throughout the timeframe. On the x-axis we have the days while the y-axis details the number of requests made.

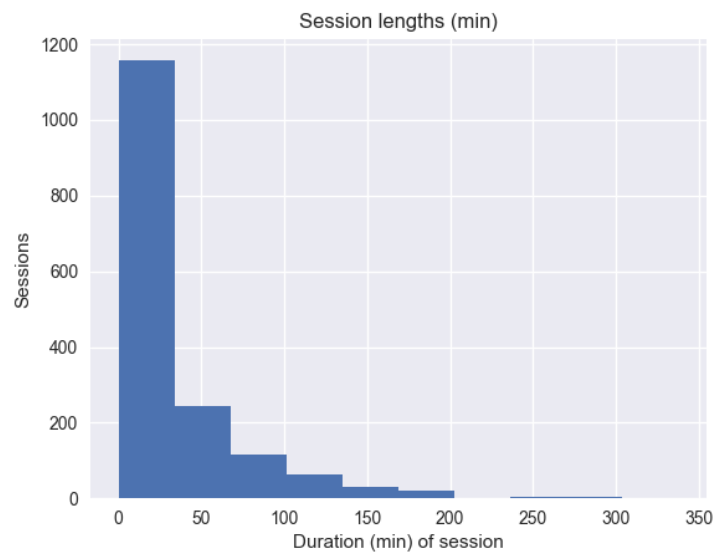






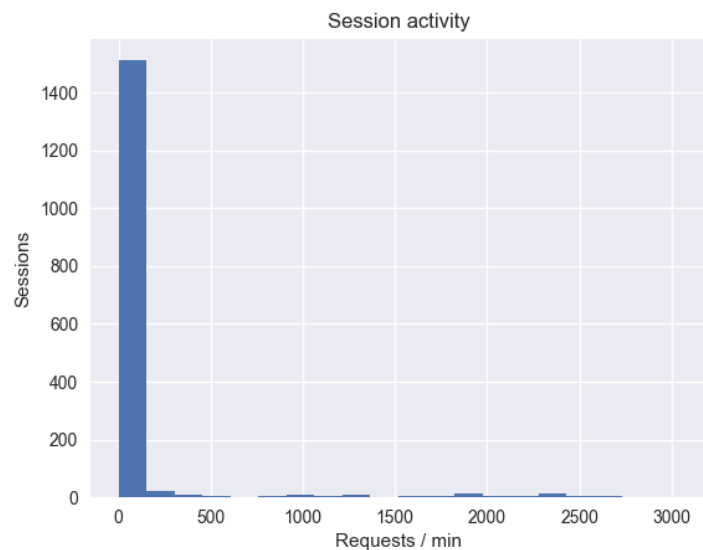
## Session lengths

The plot below details the spread of session lengths in minutes. The x-axis represents the duration of the sessions in minutes, while the y-axis details how many times sessions of this length occur. The command line flag '-sl [value]' can be used to filter out session lengths  $\geq$  value.



## Session activity

The plot below details the activity of various sessions. The activity is given by  $(\#nr\_requests/\#mins)$  for each session. The y-axis shows how many sessions correspond to each session activity value. The command line flag '-sa [value]' can be used to filter out session activity  $\geq$  value.





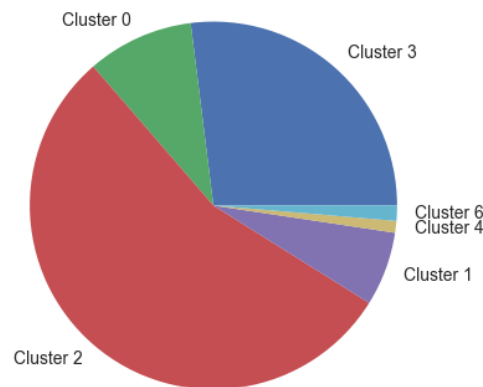
# Workflows

## Introduction

Below we can observe the distribution of sessions across the clusters. Filtered 1 cluster(s) due to small size. '-cs [value]' can be used to set the minimum cluster size.

In the following pages we dive deeper into the clusters. We provide the distribution of roles, and the tasks being performed by the users. For each of these tasks, the number of calls associated to this task were counted. These counts were then divided by the maximum count to obtain the frequency in the left column of the table.

Additionally, we provide the route that the representative session took in terms of pages. This session is known as the centroid of the cluster and has the least average distance to other sessions in the cluster.

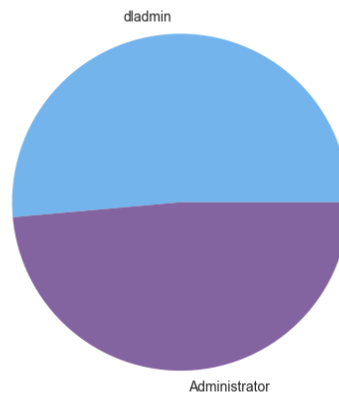




# Cluster 0

## Overview

This is an overview of cluster 0. It has a total of 154 sessions.



| Frequency           | Tasks                                 |
|---------------------|---------------------------------------|
| 1.0                 | calculations.lol.view                 |
| 0.9227539882451721  | search.search                         |
| 0.7766582703610412  | search.configurator                   |
| 0.7682619647355163  | search.vehicle.details                |
| 0.7061293031066331  | search.search.tracecar                |
| 0.49706129303106633 | quotation.quotation.list              |
| 0.35768261964735515 | calculations.calculationTable.view    |
| 0.1452560873215785  | quotation.quotationTemplate.view      |
| 0.14273719563392107 | calculations.calculationTable.content |
| 0.12762384550797648 | quotation.person.manage               |
| ...                 | ...                                   |

## Page Journey

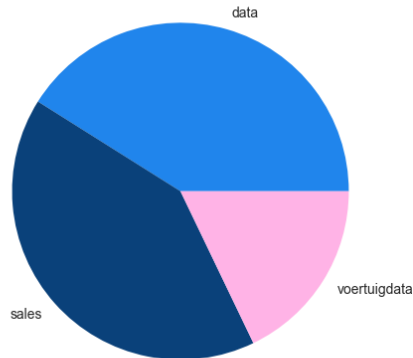
search -> quotation



# Cluster 1

## Overview

This is an overview of cluster 1. It has a total of 107 sessions.



| Frequency | Tasks                                 |
|-----------|---------------------------------------|
| 1.0       | search.search                         |
| 0.848     | search.configurator                   |
| 0.5056    | calculations.calculationTable.view    |
| 0.4744    | search.vehicle.details                |
| 0.4536    | search.search.tracecar                |
| 0.4536    | calculations.lol.view                 |
| 0.3064    | calculations.calculationTable.content |
| 0.0256    | search.list                           |
| 0.0232    | datamanagement.report.download        |

## Page Journey

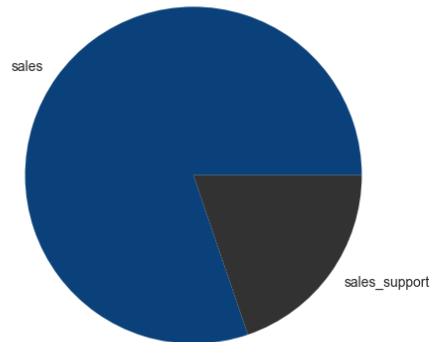
search -> table -> search -> table -> search -> configure



# Cluster 2

## Overview

This is an overview of cluster 2. It has a total of 899 sessions.



| Frequency            | Tasks                      |
|----------------------|----------------------------|
| 1.0                  | search.configurator        |
| 0.49461254612546124  | quotation.quotation.list   |
| 0.4188929889298893   | search.search              |
| 0.2908487084870849   | search.vehicle.details     |
| 0.28974169741697414  | search.search.tracecar     |
| 0.28974169741697414  | calculations.lol.view      |
| 0.22                 | quotation.quotation.create |
| 0.07225092250922509  | quotation.quotation.manage |
| 0.026125461254612545 | quotation.person.view      |

## Page Journey

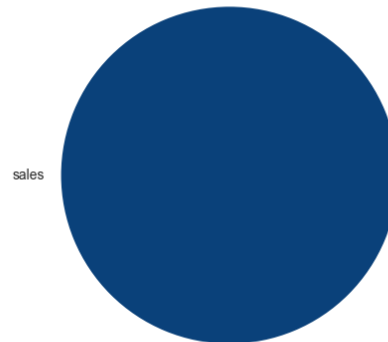
quotation\_pdf -> configure -> search -> configure -> quotation -> search -> configure -> quotation -> configure -> search -> configure -> search -> quotation -> configure -> search -> quotation -> configure -> quotation -> configure -> quotation -> search -> configure -> search -> quotation -> configure -> quotation -> configure -> quotation -> search -> quotation -> configure -> quotation -> configure -> quotation -> search -> configure -> quotation -> search -> quotation -> search -> configure -> quotation -> configure



# Cluster 3

## Overview

This is an overview of cluster 3. It has a total of 443 sessions.



| Frequency           | Tasks                      |
|---------------------|----------------------------|
| 1.0                 | search.configurator        |
| 0.35039102346140766 | quotation.quotation.create |
| 0.18344100646038763 | search.vehicle.details     |
| 0.18310098605916356 | search.search.tracecar     |
| 0.18310098605916356 | calculations.lol.view      |
| 0.13447806868412104 | quotation.quotation.list   |
| 0.1171370282216933  | quotation.quotation.manage |
| 0.08942536552193131 | search.search              |

## Page Journey

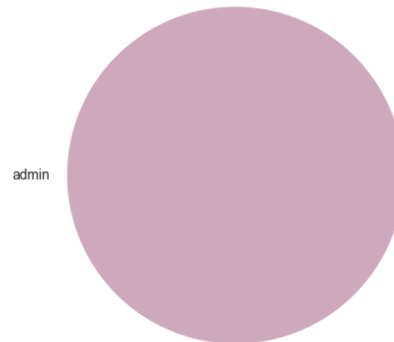
configure\_vehicle -> configure -> quotation -> search -> configure -> quotation -> configure -> quotation -> configure -> quotation



# Cluster 4

## Overview

This is an overview of cluster 4. It has a total of 17 sessions.



| Frequency           | Tasks                            |
|---------------------|----------------------------------|
| 1.0                 | search.search                    |
| 0.9056603773584906  | calculations.lol.view            |
| 0.8490566037735849  | quotation.quotation.list         |
| 0.6037735849056604  | search.vehicle.details           |
| 0.6037735849056604  | search.search.tracecar           |
| 0.32075471698113206 | quotation.quotationTemplate.view |
| 0.16981132075471697 | calculations.label.view          |
| 0.1509433962264151  | quotation.quotation.create       |
| 0.1320754716981132  | search.list                      |
| 0.11320754716981132 | search.selection                 |
| ...                 | ...                              |

## Page Journey

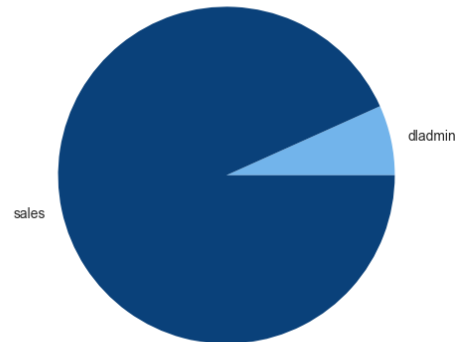
search -> label -> quotation



# Cluster 6

## Overview

This is an overview of cluster 6. It has a total of 22 sessions.



| Frequency            | Tasks                      |
|----------------------|----------------------------|
| 1.0                  | search.search              |
| 0.934010152284264    | search.configurator        |
| 0.5076142131979695   | calculations.lol.view      |
| 0.4949238578680203   | search.vehicle.details     |
| 0.4949238578680203   | search.search.tracecar     |
| 0.4593908629441624   | quotation.quotation.list   |
| 0.30710659898477155  | quotation.quotation.create |
| 0.09644670050761421  | quotation.quotation.manage |
| 0.04568527918781726  | quotation.person.view      |
| 0.015228426395939087 | quotation.relation.manage  |
| ...                  | ...                        |

## Page Journey

quotation -> configure -> search -> configure -> search -> quotation -> configure -> search -> configure -> quotation





# Page Repetitions

## Explanation

Within sessions certain set of pages are often visited in the same order. The repetitions of pages were analyzed for each session. A repetition consists of a set of at least 2 pages that occur at least 3 times within a session. Larger repetitions, consisting of a smaller repetition repeated, were filtered together with repetitions that were a rotation of another repetition. Below we observe the repetitions and in how many sessions they occurred.

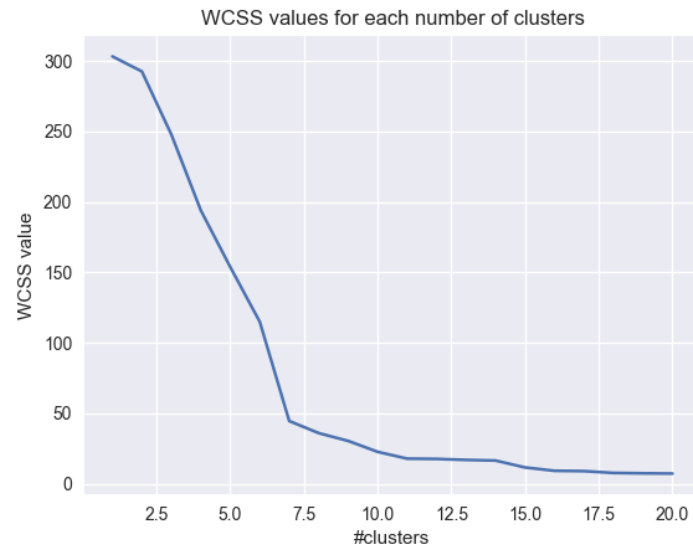
| Frequency | Repetition   |
|-----------|--|
| 503       | configure, search  |
| 467       | quotation, configure   |
| 114       | search, quotation  |
| 113       | quotation, configure, search   |
| 109       | quotation, search, configure   |
| 81        | configure, quotation, configure, search                                  |
| 59        | quotation, person  |
| 50        | configure, vehicle   |
| 40        | quotation, search, configure, quotation, configure                       |
| 34        | lol, table   |
| 33        | search, quotation, configure, quotation                                  |
| 32        | table, search, configure   |
| 26        | quotation, search, configure, search                                     |
| 26        | search, table  |
| 24        | configure, search, configure, quotation, search                          |
| 23        | search, configure, quotation, configure, quotation, configure, quotation |
| 21        | vehicle, search  |
| 21        | quotation, configure, quotation, configure, quotation, search            |
| 21        | configure, quotation, configure, quotation, configure, search            |
| 18        | search, quotation, configure, quotation, configure                       |
| ...       | ...  |



# Appendix

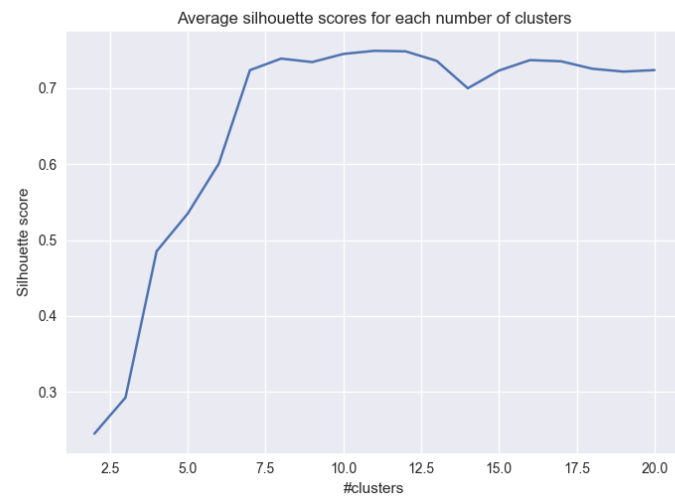
## WCSS

The optimal number of clusters for the WCSS lies in the 'elbow'. This is the point where rapid decreases cease.



## Silhouette

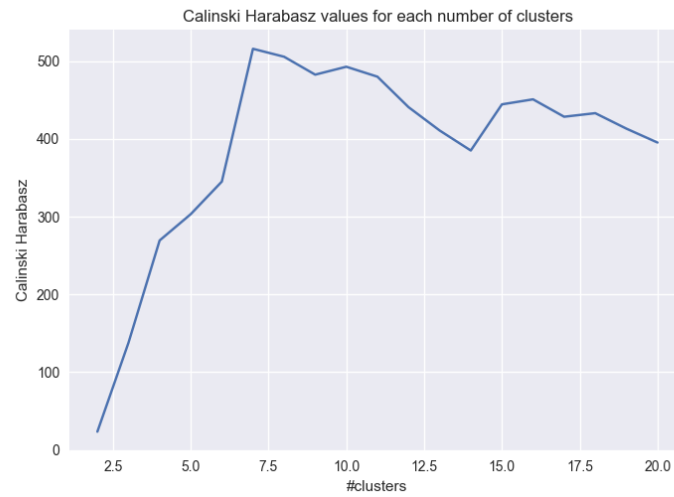
The optimal number of clusters for Silhouette lies in the highest point to maximize the score.





## Calinski-Harabasz

The optimal number of clusters for Calinski-Harabasz lies in the highest point to maximize the score.



BetterBe B.V.  
P.O. Box 1379

7500 BJ Enschede  
The Netherlands

+31 (0) 53 48 00 680  
CC no. 08097527

**Better** <> **Be**

Transforming automotive  
Leasing worldwide

© 2019 BetterBe B.V. All rights reserved. No part of this document may be reproduced in any form, by print, photo print, microfilm or any other means without written permission from BetterBe B.V.