Enabling knowledge accessibility for a customer support unit with an information retrieval portal

Ignacio Rodriguez Burgos

School of Science

Thesis submitted for examination for the degree of Master of Science in ICT and Innovation.

Espoo 28.09.2020

Supervisor

Dr. Mikko Kurimo

Advisor

MSc. Simo Säynevirta

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO

www.aalto.fi

Abstract of the master's thesis

Author Ignacio Rodriguez Burgos

Title of thesis Enabling knowledge accessibility for a customer support unit with an information retrieval portal

Programme ICT Innovation

Major Data Science **Code of Major:** SCI3095

Thesis supervisor Dr.Mikko Kurimo

Thesis advisor(s) M.Sc. Simo Saynevirta

Collaborative partner ABB oy

Date 28.09.2020 **Number of pages** 93 **Language** English

Customer support units generate massive amounts of free-text data. The retrieval of this information can be of great interest for companies to provide better customer support services. The customer support units in manufacturing companies have particular requirements in this aspect, sometimes requiring complex technical information from past cases. However, the information tools available for in this customer support units sometimes can prove to be inefficient and complicated from the workers' point of view.

This thesis explores how an information retrieval portal, built as an Minimum-Valuable-Product, provides relevant information accessibility to a customer support unit. A full-stack solution was developed after analysing the workers' information needs, knowledge tools and state-of-the-art technologies in information retrieval. The technology used stores in a No-SQL information retrieval storage system which provides a new data scheme, and a boolean scoring function is used to retrieve the documents. The solution is evaluated from two points of view: from an information retrieval point of view by measuring the precision of the first ten results retrieved, and from a user experience perspective, to measure the learnability and user satisfaction of the new tool in comparison with the old one .

As the main results of this thesis, we show that the boolean query strategy was suitable for the information needs of the users. The precision of the retrieval system achieved good results. The design of the portal provided a simple and easy user interface for users as it is reflected in the learnability results. In general, the solution was well received by the users as the commentaries provided show. Nevertheless, further improvements can be made by addressing some problems design flaws that the users flagged.

Keywords information retrieval, full-stack solution, search engine, boolean model, customer support, faceting retrieval, minimum-valuable-product, mvp, user experience, user interface

Contents

Acknowledgements

I will like to thank my advisor Simo Saynevirta for giving me the opportunity to do this project, as well as guiding me through the process. To my teammates, Antti Lukkari and Iiro Pelli, for helping me brainstorm the challenges I faced during this thesis. In addition, I would like to thank my supervisor Mikko Kurimo, for his mentorship through the whole project.

I am grateful to the Europe Institute of Technology (EIT) to give me the opportunity of accessing an international education programme adventure. I am obliged to my fellow EIT colleagues that have helped me understand the different disciplines that this project conveys. Memorable mentions are for Luca Scotton for answering countless questions regarding systems architecture and natural language processing, Adam Bako for his critical design feedback for the front-end, Rachhek Shrestha, for helping me understand React and javascript, Andrea Corsini, for his feedback regarding cloud computing and system design and lastly my love, Magdalena Mihalache, who's unconditional support during this project has helped me see the light, even in the darkest moments.

Finally, I am extremely grateful to all my family for their love, prayers, caring and sacrifices for educating and preparing me for my future. I am very thankful to my siblings, whose innumerable successes in life raised my standards to pursue excellence. To my mother, for teaching me the meaning of perseverance and sacrifice. To my uncles, for teaching me the importance of character. And finally, I am grateful to my father, for teaching me the relevance of initiative and humbleness. Let this document be a written proof of my potential to make from this world a better place.

Otaniemi, September 23, 2020

Ignacio Rodriguez Burgos

Symbols and abbreviations

CS	Customer Support
IT	Information Technologies
MVP	Minimum Valuable Product
WBCP	Web-based Collaborative Platform
CRM	Customer Response Manager
TF	Term Frequency
IDF	Inverse Document Frequency
PRP	Probability of Relevance Principle
docIDs	Document identifiers
RMUE	Remote Moderated Usability Evaluation
GDPR	General Data Protection Regulation

Chapter 1

1. Introduction

1.1 Motivation

Knowledge accessibility, storage and retrieval had been part of human beings' lives since five thousand years ago. The library of Alexandria, in Egypt, is one of the most remarkable examples of knowledge storage before Christ. Alexander the Great, who is believed to found the city and named it on his behalf, wanted to store the knowledge and works of the people he conquered as a way to preserve the knowledge from other civilizations, to be able to access to it in times of specific needs.

In our time, Knowledge management is still one of the most critical assets that companies possess. [1] In the past 50 years, the information storage methods have changed from books, images written and painted on a surface to Solid State Drives or SSD that store information by filing electrical charges in small semiconductor cells. Furthermore, the ways humans search and interact with stored information are rapidly changing. Nowadays, two or three keywords related typed in a search engine like Google or Bing are enough to retrieve millions of results related to the query. In addition to storing the information in documents and books, information has also been transferred verbally throughout history, in the form of trainings. This often happens due to the practicalities that may occur while solving a problem that can be too specific or complex to be detailed in books and documentation.

Natural language data is one of the most abundant types of data in the world; human beings generate it. This type of data cannot be processed by the same means as numerical data is processed. However, there is a subfield of computer science that oversees this, Natural Language Processing (NLP). NLP is a subfield of linguistics; the experts on this field are involved in text and speech processing and recognition, morphological and syntactic analysis, semantic analysis. This field started with the processing of symbols in the 1950s, evolved to a more statistical approach around the 1990s, and now since the 2010s, deep neural network and representation learning style methods have been ongoing [2].

Nowadays, according to Bernard Marr in [3] ,2.5 quintillion bytes of data were created each day in 2018. To access these massive amounts of information, especially human processable data, information retrieval is the technology that enables this. Information retrieval is the science of information search through any type of digital representation of information. It has the objective of retrieving this information in any format processable for human beings (text, images, sounds). Information retrieval has increased its efficiency with the development of more advanced NLP tools.

Manufacturing companies often capitalize on the knowledge they acquire while developing their products to offer customers better support service. [4] Moreover, the integration of new information technology (IT) tools in customer support units allow better services globally. Companies use these digital tools to offer this information as a service, storing it in cloud storage systems to be accessed from any part of the world subsequently. One drawback is that the amount of information gathered is immense, and the tools to order, manage and access this information can be challenging to use, provoking low efficiency in customer support units.

Usually, companies refer to customer issues as "Tickets" or "Cases". The archetype of these issues is formed by message and Case features. The message, free-text data, is created by the customer and contains information about the circumstances of the problem and demanding of the customer. The Case features consist of data attributed to the case like e.g. product model, data of ticket opening.

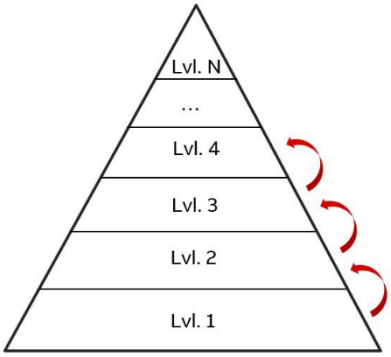

Figure 1: Archetype of a hierarchical customer support unit

global customer support units in manufacturing companies have adapted to this problem, building up a hierarchical organization through which all customer inquiries will go through. This level formation serves the purpose of filtering the less complicated customer issues. In this way, companies avoid overwhelming their customer support engineers who, because of this hierarchical system, can focus on more valuable tasks and avoid cumbersome issues. [5]

Companies generally provide support to their customers through different support channels. Each channel is leveraged following its suitability for the type of issue conveyed, which can have different levels of Urgency and Complexity [6]. Even though the lowest levels of the customer support hierarchy should be able to deal with most of the simple cases, sometimes more particularities provoke the escalation to the next level, e.g. no decision power, work saturation of the level.

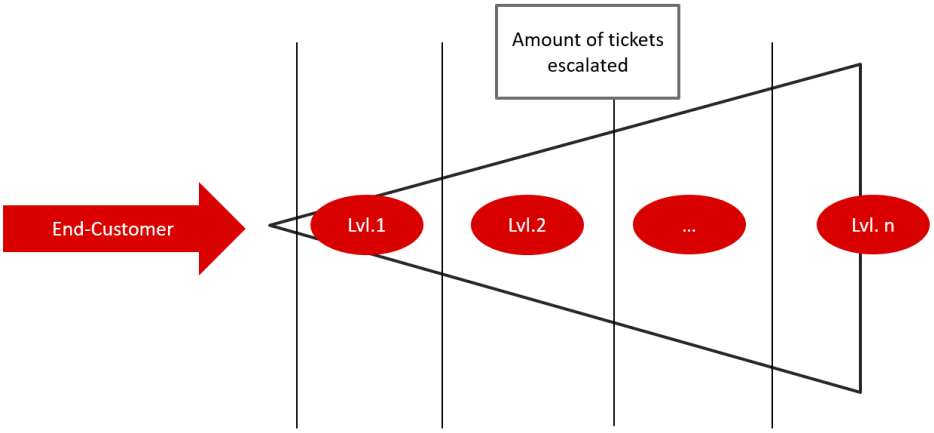

Figure 2: Visual representation of CS tickets accumulation

As this consequence cannot be avoided without interfering in the companies' decision hierarchy, this induces an accumulation of tickets in the last levels of the hierarchy. The last levels, which are also the most experts as mentioned before, must deal with a mixed type of customer tickets, burdening the unit down and making them unable to focus on complicated problems.

1.2 Research problem and goals

The objective of this thesis is to study a customer support pipeline of a manufacturing company and propose a quick and valuable solution in the form of a Minimum Valuable Product (MVP).

The main research question covered in this thesis is:

RQ1: How to find previous relevant cases from a knowledge base quickly, to speed up the resolution of the new case at hand?

To support this previous question and understand better the intricacies derived from it, the following sub-questions are addressed throughout the documents:

RQ2 What are the minimum/key features that an MVP, dedicated to fast information retrieval for customer support, has?

RQ3: What are the best scoring functions to list nested data type documents with heterogeneous features attributed to them?

RQ4: What are potential challenges encountered in building an MVP, and how can those be tackled?

1.3 Structure of the thesis

This thesis is structured in eight chapters, the first four chapters explain the investigation, study of the problem, and also provide a brief background of the topic, the following chapters describe the evaluation performed and the results obtained.

In Chapter 2, we describe an overview of the customer support pipeline. After that, the framework used for the research methodology and the research results describe an in-depth view of the pipeline.

In Chapter 3, we go through basic concepts of the information retrieval state-of-the-art. An entire information retrieval pipeline is presented, and subsequently, all the different components are explained. Furthermore, we describe the research done to find the best technology that could solve the main research problem. Once the selection is presented, the basic concepts of this technology are presented.

In Chapter 4, we describe how we build or configure all the concepts of the solution provided, and we discuss the reasons why they were chosen. In Chapter 5, we describe the different evaluation methods used. Furthermore, we describe the metrics which are used to measure the effectivity and performance of the solution.

In Chapter 6, we show the results of the evaluation performed. The results are divided into three sections. The first two sections describe the results obtained in the evaluation performed while the last section provides a global overview of the thesis.

In Chapter 7, we describe the limitations that the current implementation may attain. Additionally, we propose areas where future work can be done. In Chapter 8, we conclude the Thesis describing how the different research questions were solved and the contribution made.

Chapter 2

2. Customer Support

2.1 Overview

A comprehensive investigation of customer support is conducted. This investigation leads to a better understanding leading edge of customer support units, mainly by studying the reasoning behind changes in the industry as well as the correlations between those changes and current challenges.

What came across as a general rule in customer support was that with new technologies, services need to adapt to meet customer expectations. Ten years ago, customers expected a long response time for their tickets to be solved by the brand's customer service. Nowadays, due to rapid changes and innovations in technologies that have shown business processes can be improved, customers' expectations have also changed, an immediate response to a customer's issue being the new norm. For this reason, it is unthinkable for a company to not provide customer service without a ticketing system or live chat.

'The State of Customer Service in 2019" report done by "Hubspot.com." confirms those as mentioned above, showing that 88% of customer service professionals acknowledge, the expectations of customers have increased dramatically in comparison with the past. Also, 76% of service professionals agree that in comparison with the past, customers show a smarter and informed attitude when communicating with them. [7]

Customer service has become a relevant subject, especially for startups who understand that to grow faster, happy customers will act as a catalyst, spreading the word about the excellent treatment received while encountering a problem. Classical manufacturing companies, mainly those focused on quality, can find inspiration in these new methodologies of customer service to adapt to future challenges. The importance of treating customer competently can define the success of the company. For this reason, customer service engineers should be appropriately equipped, trained, and motivated to deliver excellent customer service.

For measuring customer experience, companies rely on different metrics to understand how good their customer service is delivered. What is essential when tracking metrics?, according to the Evidence-Based Management framework, a famous framework for tracking a company's success, is to measure the right value to the business. This framework shows that value can be added through 4 key areas:

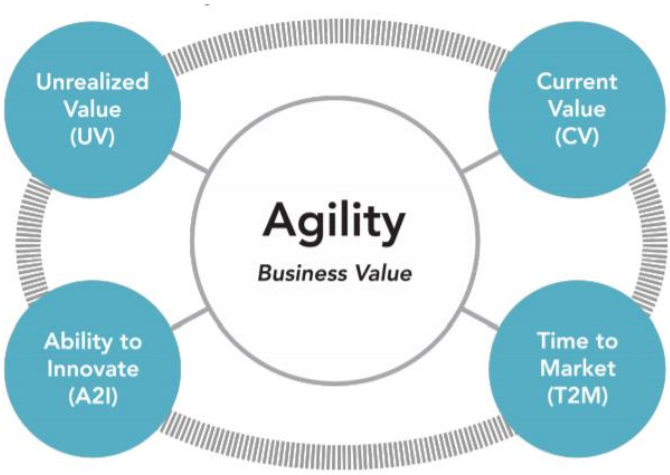

Figure 3:Evidence-Based Management framework

One of the most relevant metrics to track the Current Value of an organization is the Net Promoter Score or NPS. NPS is a tool that proposes to measure customer loyalty by categorizing customers based on their responses to the question: "How likely would you recommend this product or service to a friend?". [6]. Many companies use NPS, and for this reason, is one of the most relevant metrics. [7]. Many other metrics can be used to showcase customer support effectiveness (resolution rate, first, response time) and all are related to how efficient customer support engineers find information and use it.

The tools that customer service engineers are equipped with have also dramatically improved to support them being prepared. The methods that the customer service engineers used to retrieve information have evolved: from asking a colleague what is the best way to solve an issue, to googling it on the Internet, using social media [8], to having a software application specialized for this purpose or even a chatbot that would answer predefined customer questions [9]. Each method comes with its advantages and drawbacks and requires a different level of training [10].

The uncertainty of knowing what metric to use to track customer success and the variety of tools that can be used to retrieve customer information shows how volatile customer experience is. For the best results in customer experience, instead of proposing a general solution for improving information retrieval in customer support, one needs to study the differences of a company since each company has different pipelines, hierarchy and tools that require a specific fit to them.

2.1.1 Study of the pipeline

Throughout this document, a specific company unit will be used as an example of showcasing why every company needs a personalized tool that matches their values, organizational structure, and culture of work.

The pipeline of the customer support needed to be studied for a better understanding of the internal processes and needs in preparation of implementing a potential solution for the topic of knowledge extraction in customer support. An introduction and general overview are presented to understand the basic structure of the unit and how the unit usually performs.

The priorly mentioned unit is therefore explained here. This unit offers support to the global business units and is the third and last step of the customer support response of the company. To support the understanding of the problem below follows an analysis of the customer support pipeline.

As it was mentioned before, customer support units are divided into different levels. In this case, the customer support pipeline is divided into three levels:

- Level 1: More commercial, front-line support
- Level 2: Both technical and commercial
- Level 3: Deep technical knowledge, provides support to the first levels and deals with significant and complex customer issues.

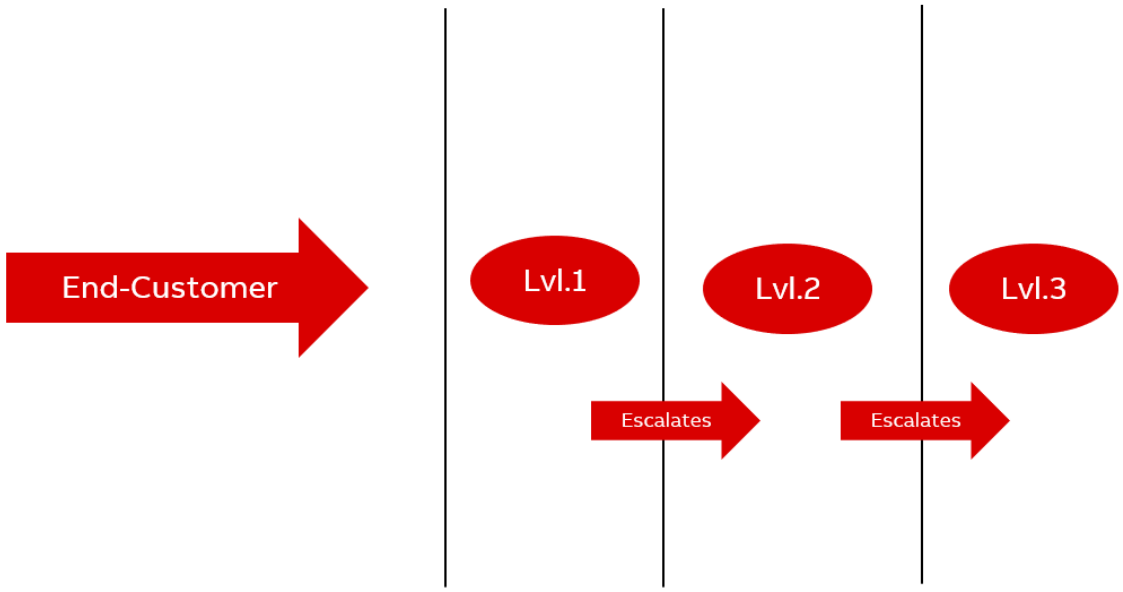

Figure 4: Pipeline of the CS unit understudy

The pipeline is structured in the following manner:

End-customers contact the first level, which will potentially try to solve the issue. If the first level does not succeed, the issue will be escalated to the second level, which generally has a less commercial and more technical focus. The second level is responsible for attempting to solve the problem, finding the information necessary to solve it. If the problem happens to be more complex, they escalate the issue to the third level. The last level is composed of engineers with in-depth technical knowledge who are recurrently trained with the latest technological advances being implemented in the company's products.

2.2 Research methodology

To provide a further understanding of the pipeline, particularly of the perspective of the direct stakeholders involved in it, research was conducted in the unit. The research was made to uncover both the qualitative and quantitative aspects of customer support through the PACT framework [11]. This framework is not new; it has been used in [12] to help design human-computer interfaces. It motivates the designer to observe the problem from 4 different lenses:

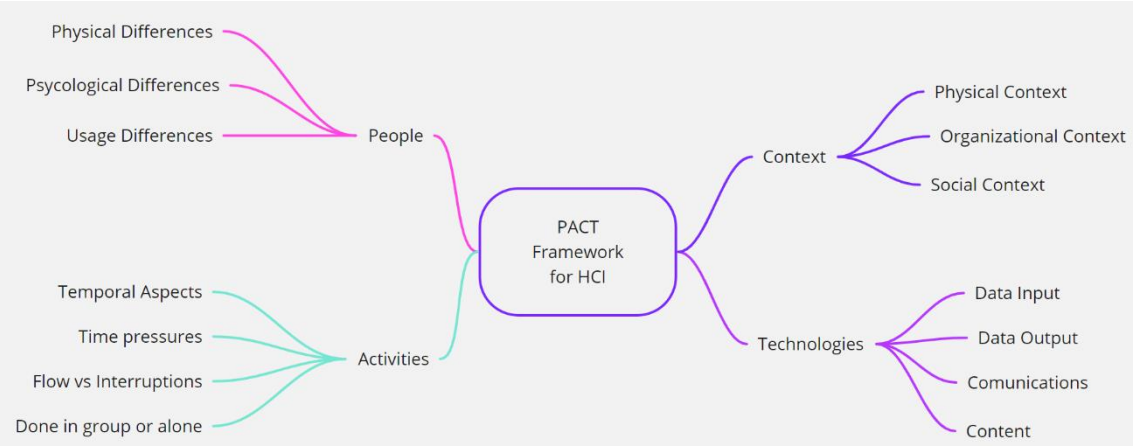

Figure 5:PACT framework mindmap

People

Who is going to use the design? It is essential to have in mind how different clusters of people are involved in the matter(problem) and how their physical, psychological and usage differences impact the problem, thus having reverberating effects on the solution. In this way, technology ceases to be used for the sake of technology, but it is relatively leveraged to support and solve real human needs.

Activities

What are the activities that the stakeholders need to be involved? This question does not expect only easy answers but also covers lengthy and tedious activities that pose a complex threat in information retrieval. The designer needs to analyze it taking into account the temporal aspect, time pressure of the activities, action interruption, response time of the activities, if the activity is conducted in groups or on an individual level and last, but not least if it can be retrieved more efficiently.

Contexts

The designer shall consider the physical context, such as the weather, temperature, or the geographical situation. The organizational context, as if the solutions will be used for indoors or outdoors activities. The Social context, so having to take into consideration how the working environment is if there is a supportive environment in which coworkers support each other or not. Specific social structures and norms can promote the acceptance or discourage of a different variety of designs that the designer needs to in mind to perceive the full picture.

Technologies

This perspective pushes the designer to consider input channels and output channels, how the communication is happening besides the characteristics concerning them. Additionally, the data content is also considering if the content is relevant, updated and well preserved.

This framework has proven to provide a useful and comprehensive overview of the problem in the design of an Online Cooperative Programming platform [12] and the design of a Health care information system [13].

For information gathering, contextual inquiries were performed to obtain information about the different points of view that PACT framework needs for the analysis to be performed. A contextual inquiry is an interview method with a semi-structured nature. It is used to obtain information about the context where the users work by first asking a series of standard questions. Afterwards, they are observed and asked while working in their environment [14].

The questions used in the contextual inquiries:

- What are the main tasks/steps to attend a Customer Support Ticket?
- Do you identify any repetitive tasks?
- What do you feel about the tools that you are using?
- How would you describe the ideal way of solving a customer ticket?
- When the customer problem is difficult for you, or you simply lack the knowledge, where do you find the information you need to solve it?
- How often do the problems that need to be solved require of teamwork?
- Does it often take long to solve users' ticket completely?
- Which are the ticket categories that need more time to be solved?

The contextual inquiries were conducted with five representatives of the customer support unit; they are the most knowledgeable individuals of the unit and have a great experience with how the unit works internally. The representatives showed a proactive attitude and a will to participate in the interviews as they express their interest in providing new perspectives for future change to the unit.

A **survey** was conducted to understand the essential features of the dataset. The objective was to generate a quantitative measure to understand which features are more important for customer support engineers. The survey consisted of a grading list where surveyees had to grade on a scale of 1-10 how useful/valuable is a feature for them in their daily tasks. A comment section for each feature was also provided if they wanted to add extra information.

2.3 Research results

In this section, the results for all the research done are presented along with personal observations that were made during the entire research process.

2.3.1 Contextual Inquiry

As mentioned in Chapter 2.1.1. *Study of the pipeline,* a percentage of customer support tickets are escalated to the next level. During interviews with CS engineers from the first and second levels, it was found out that the main reasons for issue escalation are lack of knowledge and lack of decision power.

A note to be taken is that there are mechanisms during the customer ticketing pipeline system to detect if an issue must be escalated directly to a determined level. Additionally, there is a general concern about a peak of issues escalated to the last levels. The accumulation of ticket is provoking a delayed response time and can potentially influence burnout of CS unit engineers.

Interviews and analysis done through the unit indicated a lack of knowledge access. Knowledge search and retrieval should improve the workflow by shortening the time it takes for a customer service engineer to find valuable information to deal with a customer case at hand.

The reasons this happened happen for different factors that were gathered through the interviews and structured afterwards:

External factors identified:

- Seasonal technical problems: These are those problems which their recurrence happen to be bound to a seasonality pattern, the reasons behind can be related to product cycle or weather exposure.
- Annual workload unbalances: These are issues escalated which are not bound to any seasonality but help to the accumulation of issues.

Internal factors identified:

- Variety of tools: CS unit has many information tools, which they need to consult based on different conditions. This situation can result in time loss by the engineer to try to find the correct piece of information.

As part of the study, the responsibilities of the regular CS engineer were distinguished by increasing the granularity of the study, getting a deep understanding of the different steps the engineer have to go through, and at the same time, identifying the steps in which is more impactful and reliable to provide an enhancement.

The CS engineer has to go through 5 phases:

1. **Ticket filtering and selection:** In this stage, the CS engineer selects the case that is going to work with, this selection can be based on ticket characteristics that the engineer is familiar.
2. **Problem understanding:** The stage in which the engineer assimilates the problem.
3. **Information gathering:** In this stage, the engineer needs to find the different pieces of information to provide a solution. This solution can be provided by sending replacement materials, teaching a specific way to fix a problem or confirming the warranty of the product is still active.
4. **Customer communication:** The CS engineer starts communicating with the customer, asking for further information if needed or finally informing the solution for the issue raised.

5. **Problem solved:** Final stage, in which the customer confirms the problem has been solved and further discussions concerning payments and documentation are arranged.

With all, it was evaluated that the first two stages could not be enhanced directly for the engineers. However, as it is mentioned in the future discussion, automatization of the ticket filtering and selection could be implemented. (talk about automatic ticket classification, CS profiler for best tickets)

Therefore, let us focus on **3) Information gathering**. The information-gathering process by engineers is done primarily by peer consulting, so the more knowledgeable and expert engineers in the CS unit concentrate experience and other unit members may consult them. The second step is to consult the Information tools available, but as it was mentioned before, there is a wide variety of tools that the engineer can consult. Nevertheless, there are two specific tools which concentrate a significant amount of information from past customer issues or "Cases".

The first tool is a web-based collaborative platform (WBCP) that works as a document management and storage system. The second tool is a cloud-based customer relationship management service or CRM. Both platforms have old cases with emails exchanged by customers and customer support engineers.

The contextual inquiry was focused on following the users while using their information tools previously described to understand the users' information needs. During this process, different observations where made:

1. The users seek to find the history of problems that happened to an old specific machine; this search is typically done using the serial number that the client provides.
2. The users need to find for a machine model, what are the most common problems; this search is too tedious as the user need to navigate and read extensive amounts of irrelevant data.
3. The users seek to find for a specific machine and problem code, a list of old cases in the past.
4. The users need to read the emails to understand what happened for each case.

These observations were made after the five contextual inquiries were completed. Since the number of users proved is low, it might be possible that more observations could have been done.

Even though these two tools concentrate significant amounts of information, the accessibility to this tool has been reported to be tedious during the interviews. Users also expressed they lose time going through irrelevant data until they can find something relevant or suitable for their customer ticket in hand.

Therefore, user satisfaction information gathered from the contextual inquiry showed that the main reasons for the users' dissatisfaction with the current tools are:

- An overcomplicated user-interface, which make it tedious to work with.
- Time-consuming tools
- Frustrations derived from searching through irrelevant data

This combination is troublesome, a tool perceived by the workers as complicated, tedious and time-consuming can lead to user's burnout. As mentioned in the Overview of this chapter, a company's customer support burnout should be of utmost concern as the customer support unit is the front-line of the public relationships with the customer.

An inspection of the current knowledge tools was done to understand why this is happening. This inspection revealed that the tools been used are not meant for information retrieval, but storage and

customer contact services. However, since contextual inquiries showed that the information relevant for the CS engineers is the information exchanged between customer and company about the problem, and the actions performed to solve it, these tools do not provide and fast and user-friendly way of doing it.

For this thesis, access to the web-based collaborative platform database was provided, and the MVP will be done based on the data of this platform. In order to not reveal sensitive information about the tool, the name and details of this tool will not be mentioned directly but will be changed. Nevertheless, the nature of them will be explained for the reader to understand the decisions made during the process of this thesis. From now on, the database will be given the acronym of WBCP database following the initials letters of "web-based collaborative platform".

Chapter 3

3. Information Retrieval

3.1 Key concepts

This section aims to give a basic notion of what information retrieval is to be able to understand the solution with a well-funded base. Information retrieval engines have a standard archetype structure as all have the same general function. Based on a query inputted by the user, the engine creates a list of documents shown in a descendent order of the ranking score.

Therefore, the objective is to retrieve those documents that are more relevant to the user's information needs and, in our case, support the CS engineers to complete a task more efficiently as the information accessibility has provided knowledge to what is doing [15]. Thus, the final objective in information retrieval is about translating the user's need into a query that can be inputted in the search box.

Classical researchers in the field of information retrieval suggest that when developing an information retrieval engine, it is of capital interest to understand if the results are satisfactory and the way to understand if they are so is by using two measures types [16]:

- Precision is the proportion of the retrieved files by the engine that are relevant to the user's information requirements, the output being the division of valuable files out of the total number of files.
- Recall is the reciprocal measurement, answers to how much of the good or useful information of the system is succeeding and finding it for the user.

However, in practice, user queries provide thousands of relevant results, and a small number of users will be interested in all of them. This fact has induced the metric recall to no be relevant anymore in modern information retrieval [17]. Therefore, Precision at k documents (P@K) is a more popular metric used in the current state of the art of information retrieval engines. For example, P@10 corresponds to the number of relevant documents from the first ten documents retrieved.

The methodology to apply is thinking about user's information need and how results precision assessed relative to that. Is of vital importance to understand the user's knowledge of the information need. This information can provide an understanding of the user's query habits (if they are prone to commit errors or if their information needs can be generalized). With all of this in mind, creating a relevance model personalize to the users need can provide better results in the long term.

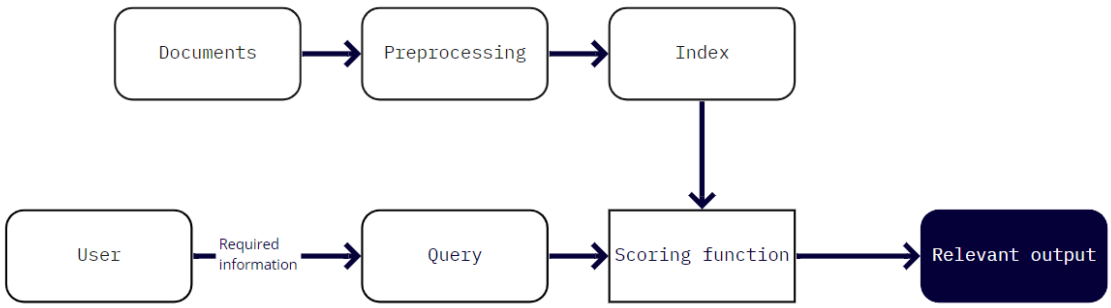

Figure 6: Diagram of an information retrieval system

After the user sends a query using the User Interface, this query is parsed to measure the similarity of the query with the documents in the database, in order to do this fast and efficiently, documents

have been previously pre-processed and indexed in a data format called inverted index. Once the ranking is done, a descending ranking of documents is outputted to the user's interface.

In the following section, a breakdown of each of these components can be found:

3.1.1 Preprocessing

In information retrieval, documents are represented as a bag of words. This words will be indexed to be compared with a query inputted by a user potentially. The problem is that vocabulary mismatched can happen, which can bias the comparison, for example, "USA" and "U.S.A." mean the same for a human being but a computer is not able to understand the difference directly. General causes of vocabulary mismatch are:

- Spelling variation and errors: "Ignacio = Icnacio = Ignachio = Ignatius"
- Morphological variation: "Woman = women ≠ man ≠ mania = maniatic"
- Word boundary variation: USA ≠ U.S.A."
- Synonymy and polysemy: "apple ≠ Apple ≠ Big Apple = New York City ≠ York City"

These issues motivated the need for a preprocessing step to normalize the data into a searchable format:

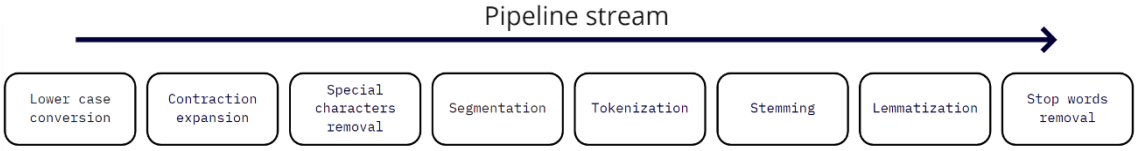

Figure 7: Preprocessing pipeline of an Information Retrieval System

- Lower case conversion: A simple step that consists of turning all characters to lowercase: "Woman -> woman."
- Contraction expansion: In a language like English or French, words can be contracted. This task generates the expanded version. "I'd -> I would"
- Special characters removal: Symbols and special characters are removed to remove the extra noise that it produces on unstructured text data. "+ ^ - *" are examples of special characters.
- Segmentation: These tasks consist of dividing texts into individual sentences. It is not a difficult task to perform as the limits of a sentence are settle by points.
- Tokenization: Once the segmentation is completed, tokenization is used in most of the cases. The tasks divide sets of sentences in sets of words, as these are the most meaningful text units.
- Stemming: The most complicated step. It aims to reduce the word to a unit that is possible to use as an indexing unit in a search engine. A good stemmer is nothing but a vast set of refined conditions. There are very common known stemmers that have been using in the Natural Language Processing field for a long time:
    - Porter Stemmer or Snowball Stemmer, which is a sequence of suffix-stripping rules stemmer. Produces stems with some mistakes
    - Lancaster Stemmer, which also is rule-based but is more aggressive in that will make more mistakes trading off with faster execution time.
    - Krovetz Stemmer: ruled-based stemmer with a broad exception list that is used to find out if the suffix from a word is considered an exception.

  An example of stemming would be words like: "jumps, jumped, jumping" would be reduced to "jump".

- Lemmatization: Similar to stemming, this step tries to extract the lemma of the word. The difference is that stemming does not take into account the context of the word. Typically, these algorithms are based on dictionary lookup algorithms, and for this reason, stemming is a less expensive computational step. For example, the word "better" would be extracted as "good".
- Stop words removal: Stop words are the function words, they link ideas, but they carry no meaning on their own. Stop words are the most frequent words in a text and hence, take large proportions of the index memory if not removed. In practice, most search engines have a stop list, and that is a list of words that are considered to be meaningless and can be removed.

In this chapter, a description of a preprocessing pipeline has been explained. This description sets the archetype from which a more domain-specific preprocessing pipeline can be generated. After going through this pipeline, the data will be ready to be stored.

3.1.2 Ranking: similarity models

Boolean model

This model is the most fundamental, and the precursor of all the ranking models in the state-of-the-art nowadays. Its main task is to check whether a document matches a query or not, returning 1 and 0 respectively. As the evaluation criteria are rigid, there is no space for ranking distribution, and documents can only have two categories. For this reason, boolean models do not satisfy precisely search requests for text documents, since doing that might result in an undesired high amount of solutions. The latter is also a reason why boolean models perform better with small pools of documents.

An example of a boolean ranking function would be the Jaccard coefficient, which is a commonly used measure of the overlap of two sets A and B:

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|} \qquad (1)$$

Which basically can be obtained by taking the item number of the intersection of A and B and divide it by the item number of the union. The main issue is, it does not consider term frequency, which causes rare terms in a collection to be more informative than frequent terms and penalizes lengthy documents over shorter ones.

Nevertheless, when user queries and information needs are based on keywords, the boolean model outperforms other types of models for its simplicity. The disadvantage is that all terms are weighted the same, so it needs some fine-tuning to be added, like adding weights to certain types of keyword families to make them more relevant [16].

Vector space model

In vector space models, the objective is to represent everything in high-dimensional space: words, documents, queries can be represented if they are transformed into a vector.

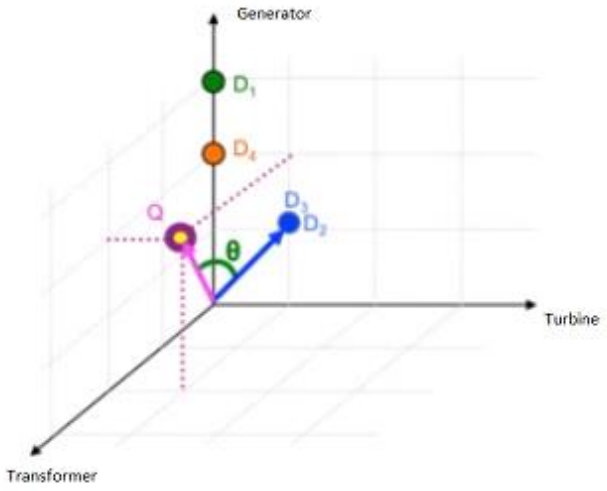

Figure 8: Four documents D and a query Q represented in a vector space

As an example, four documents D are represented and query Q:

Doc1: "Generator Generator Generator" -> (0,3,0)

Doc2: "Generator Turbine" -> (0,1,1)

Doc3: "Turbine Generator" -> (0,1,1)

Doc4: "Generator Generator" -> (0,2,0)

Q: Generator Generator Turbine Transformer Transformer -> (2,2,1)

The Euclidean distance is calculated to evaluate the similarity between the Q and the D, and assuming unit-length vectors, the angle between the vectors is measured. This is represented using the dot product operator.

$$s(Q,D) = \sum_{w} Q_w \cdot D_w \qquad (2)$$

Were Q_w are the weight of the word w on in the query Q and D_w is the weight of the word w in the document D. As it needs to take into consideration the fact that keywords tend to be repeated in a doc and avoid bias results for long documents D, the similarity function operators can be transformed to:

$$s(Q,D) = \sum_{w} tf_{w,Q} \cdot \frac{tf_{w,D}}{|D|} \qquad (3)$$

Where:

- $tf_{w,Q}$ is the number of times the word w occurred in Q
- $\frac{tf_{w,D}}{|D|}$ is the **Term frequency component** or (tf) is the number of times the word w occurred in D normalized by the document length $|\ D\ |$

This last component introduces the notion of term frequency in weighting the document and query and calculating the similarity between them. However, this similarity function does not promote rare words, which generally carry more meaning. As KS Jones states in [18] "*But it seems we should treat matches on non-frequent terms as more valuable than ones on frequent terms, without disregarding the latter altogether*".

For this purpose, the document weight operator will now be multiplied by the (idf) or **inverse document frequency** factor: $\log \frac{|C|}{df_w}$.

$$s(Q,D) = \sum_w tf_{w,Q} \cdot \frac{tf_{w,D}}{|D|} \cdot \log \frac{|C|}{df_w} \qquad (4)$$

Were $|C|$ is the number of documents in the collection and df_w is the number of documents containing w, the log operator is used to even the scale with the *tf* component. idf is a handy heuristic for selecting out essential words, sometimes is used on the query weights Q_w.

Nevertheless, this similarity function does not perform well enough, as it can be observed in the following example:

- User inputs the query Q = "Azipod propeller."
- Documents collection = {D1= "…Azipod…..propeller", D2="…. Azipod …. Azipod …."

The user most probable will like to receive D1 as output, but the current version of the similarity function $s(Q,D)$, spec ranks higher the documents with rare words present in the query Q. So for this case, D2 would be ranked higher because idf operator is higher for D2 than D1. Idf has this unfortunate property provoking a document literate high idf words (rare words) will be boosted up.

To correct this, a diminishing returns growth parameter k is added to the term frequency, so the first occurrence of a word is more important, and the following occurrences of the word are less critical [19]. Nevertheless, this diminishing returns effect is not always right, specifically in long documents, because if in this documents a word that occurs once has great relevance if it appears more times it reinforces the relevance of this word in the document. So make k depend on the length of the document $|D|$, so for long documents k will have big values and for short documents, repetitions are not that important.

$$s(Q,D) = \sum_w tf_{w,Q} \cdot \frac{tf_{w,D}}{tf_{w,D} + \frac{k|D|}{avg|D|}} \cdot \log \frac{|C|}{df_w} \qquad (5)$$

$$s(Q,D) = \underbrace{\sum_w tf_{w,Q}}_{Q_w} \cdot \underbrace{\frac{tf_{w,D}}{tf_{w,D} + \frac{k|D|}{avg|D|}}}_{\text{tf component}} \cdot \underbrace{\log \frac{|C|}{df_w}}_{\text{idf component}} \quad \text{(with } D_w \text{ over the tf component)}$$

Figure 9: Visual reference of the different components of the scoring function

k: is to be tuned, different sets of documents use different values to get better performance out of the similarity function.

So we have obtained a state-of-the-art ranking formula for ranking documents in response to short queries. Variations of this formula are used actively by many search engines nowadays. Nevertheless, to search for big queries in essential documents or even compare documents, cosine similarity is the most used vector similarity measure is one of the most famous function used:

$$s(Q,D) = \frac{\sum_w Q_w D_w}{\sqrt{\sum_w Q_w^2} \cdot \sqrt{\sum_w D_w^2}} \quad (6)$$

Where the weights of the vector Q_w and D_w are:

$$Q_w = \frac{tf_{w,Q}}{tf_{w,Q} + \frac{k|D|}{avg|D|}} \cdot \log \frac{|C|}{df_w} \quad (7)$$

$$D_w = \frac{tf_{w,D}}{tf_{w,D} + \frac{k|D|}{avg|D|}} \cdot \log \frac{|C|}{df_w} \quad (8)$$

The vector space model has shown to be a decent model for search engines; it is very versatile as it accepts various types of weights. It has a relevant reputation, but by itself, it does not suggest what to try next, in other words, is too general. This gap is where probabilistic models take place.

Probabilistic model

Theses model ranks document by the probability of user relevance, based on the data available on the system. Theoretically, these models can estimate accurately "on the bases of whatever data have been available to the system for this purpose", which means that these models will need to use all the data available in the system, documents, sessions, context, user profile, to be able to provide the best results.

This is called the Probability of Relevance Principle and was first mentioned by Robertson and Sparck-Jönes in 1977 [20].

The Probability of Relevance Principle (PRP) says that if one takes the documents and rank them by the posterior probability of relevance, the best ranking possible will be obtained. By this claim, the PRP gives optimal values concerning precision/recall at a given rank and average precision. All these claims may sound promising, but the real problem is in how this is estimated, the classical attempt by Robertson and Sparck-Jönes is also known as the Binary Independence model which culminates in the BM25 ranking model, is the most influential piece of work in this area.

The best way to understand how this model work is by understanding how it was derivated. This classical probabilistic model is based on different assumptions, each one helping to transform the formula until a final version is obtained [16].

Let us take a look at this claim, let $D_i = \{Document_i, Query, User, Task, Context, \dots\}$ be our representation of documents, let $p_i = P(R_i = 1|D_i)$ the posterior probability that a document D_i is relevant. The first assumption A0 is:

- *A0*: The relevance of document D does not depend on any other document. As the objective is ranking documents by the probability of relevance, this can be written as the probability observing a document given that under the relevant class divided by the probability of observing a document given that is not.

$$P(R = 1|D) = \frac{P(D|R = 1)}{P(D|R = 0)} \qquad (9)$$

- A1: Words in a document shall be absent or present, which means that a document is a collection of Bernoulli (binary values) values, with 1s for word occurring in the document and 0s for words not occurring in the document.

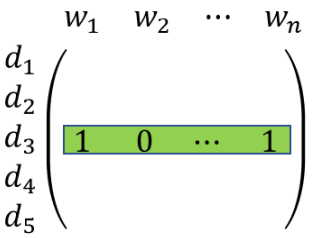

Figure 10:Bernoulli representation of a document.

- A2: All the representations of words D_w are mutually independent. This assumption is very strong but necessary as the event space is so huge. Additionally, as our documents are binary vectors, then documents are subsets of vocabulary, and our vocabulary can get quite large. Furthermore, there are lots of subsets of such a vocabulary that prevent us from estimating anything correctly, unless an assumption is made that allows us to factor the probability in some way. However, if the assumption is made, the equation becomes more straightforward, as per below:

$$P(R = 1|D) = \frac{P(D|R = 1)}{P(D|R = 0)} = \frac{\prod_w P(D_w|R = 1)}{\prod_w P(D_w|R = 0)} \qquad (10)$$

At this point, it can be defined $p_w = P(D_w = 1|R = 1)$ as the probability that the world w is present given the document is relevant and $q_w = P(D_w = 1|R = 0)$ is the probability that the word w is present given the document is not relevant.

- A3: the probability of an empty document being relevant has the same probability under the relevant class and the non-relevant class, $P(\vec{0}|R = 1) = P(\vec{0}|R = 0)$

$$P(R = 1|D) = \frac{\prod_w P(D_w|R = 1)}{\prod_w P(D_w|R = 0)} = \frac{\prod_w \begin{pmatrix} p_w & \cdots w \in D \\ 1 - p_w & \cdots \notin D \end{pmatrix}}{\prod_w \begin{pmatrix} q_w & \cdots w \in D \\ 1 - q_w & \cdots \notin D \end{pmatrix}} \qquad (11)$$

The products in equation(number) goes over all the words, for the words that occur in the document, p_w and q_w is used, for the words that not occur in the document, $1 - p_w$ and $1 - q_w$ is used. This formula can be reordered to obtain:

$$P(R = 1|D) = \frac{\prod_{w \in D}\left(\frac{p_w}{q_w}\right)\prod_{w \notin D}\left(\frac{1-p_w}{1-q_w}\right)}{\prod_{w}\left(\frac{1-p_w}{1-q_w}\right)} \quad (12)$$

Where the first product in the numerator is the ratio of the probability of the word that occurs in a document, the second product is the ratio of the probability of a word not occurring in a document the denominator product is the ratio of the probabilities that documents are empty. As it was just assumed in A3, this last product yields one, so the final form obtained is:

$$P(R = 1|D) = \prod_{w \in D}\frac{p_w(1-q_w)}{q_w(1-p_w)} \quad (13)$$

The interpretation of this is a trick to go from one product that runs over the entire vocabulary, to a product over just the words in the document, and this allows faster computational runs as the model will be dealing with a small % of total vocabulary and will allow term-at-a-time execution..

- A4: If the word w is not in the query, it is equally likely to occur in relevant and non-relevant populations.

$$p_w = q_w \ldots w \notin Q \quad (14)$$

The practical reason for this assumption is to help restrict the product to query/document overlap:

$$P(R = 1|D) = \prod_{w \in D \cap Q}\frac{p_w(1-q_w)}{q_w(1-p_w)} \quad (15)$$

- A5: If the word does occur in the query, it will occur in half the relevant documents on average. The practical reason is to cancel out p_w and $(1 - p_w)$.

$$P(R = 1|D) = \prod_{w \in D \cap Q}\frac{(1-q_w)}{q_w} \quad (16)$$

- A6: For any given query, almost the entire collection is non-relevant; only a small fraction will be relevant.

So, if N_w is the number of documents that contain the world w and N the total number of documents we can approximate to:

$$q_w \approx \frac{N_w}{N} = \frac{N_w + 0.5}{N + 1} \quad (17)$$

And from equation 16 is obtained:

$$P(R = 1|D) = \prod_{w \in D \cap Q} \frac{1 - N_w + 0.5}{N_w + 0.5} \quad (18)$$

Looking close to this formula, if logarithm would be applied, idf term would be obtained. The whole formula boils down to a sum of the IDF values for the query term in the document.

These seven assumptions give space to produce the classical probabilistic model. Numerous researchers have tried to improve this formula by rescoping one of the assumptions.

An assumption that is continuously being modelled is the A2; Classical model assumes all words independent, and an example of this is Van Rijsbergen with a tree dependence model [21]. The idea was to model dependencies between words by forming a maximum spanning tree that computed the probability of a query based on the probability of a word being close to its parent in the tree.

The other assumption that has been tried to model is A1, which is related to word frequencies modelling. The idea presented by Harter was that a mixture of two Poissons distribution generates words in a document, were "elite" words for a document occur with unusually frequency and "non-elite" words occur by chance.

Eliteness is a variable that will dictate whether a word is going to come from the heavy head or heavy tail of a Poisson distribution. Robertson, Spark-jones did this idea. The authors of the original PRP, come up with an approximation to the 2-poisson called the BM25:

$$\log \frac{P(D|R = 1)}{P(D|R = 0)} \approx \sum_w \left(\frac{d_w(1 + k1)}{d_w + k1\left((1 - b) + \frac{b \cdot dl}{avg\ dl}\right)} \cdot \log \frac{1 - N_w + 0.5}{N_w + 0.5} \right) \quad (19)$$

The logarithm of a ratio of two documents probabilities is just the sum of the words of that quantity which they came out as an approximation of the probability ratio of a given the word. Looking closely to the formula, one can identify some common features with equation 5 from vector space models:

- The first operand from the product is the TF term and the second is the IDF
- The sum operator can be interpreted as the more words in common with the query the better results obtained
- The TF term makes query words repetitions have more chances, while repetitions of the same words have fewer chances than query words.
- $k1$ and b are parameters to be tuned, with $k1 \in [1.2,2.0]$ and $b = 0.75$, according to [16] p.233.
- $\frac{b \cdot dl}{avg\ dl}$ normalizes document length as it was done in equation 5
- $\log \frac{1 - N_w + 0.5}{N_w + 0.5}$ is the IDF component, which makes common words less important

To wrap up, the PRP establish a framework in which ranking documents by the probability of relevance is optimal. After going through different assumptions that the creators of the classical probabilistic model did, it has been shown how this model is quite hermetic, does not give too much space for modelling relevance or frequencies. Therefore, different transformations through time have tried to enhance this model, leading up to approximations that better fit the information retrieval framework. This is the BM25, Okapi BM25, named before the system where it was tried.

Conclusion about the similarity models

The three information retrieval models were explained in this chapter to give a clear view of how they work.

The boolean model is useful for its simplicity. On the other hand, it brings some problems for similarity measurement as it can be True or False. This fact makes it useful when users want to search for unique identifiers or keywords, in other words, data that should be or not there. For example, if a user is looking for the unique identifier 64, it is irrelevant for the user to receive documents with a unique identifier of 65.

The vector space model provides an advanced document representation, it takes into account term frequency (tf) in a document as a general concept but also takes into account how non-repetitive words bring relevance to a document with the inverse document frequency or idf. This model has different drawbacks:

- It scores long documents poorly as the scalar product scores small results due to the high dimensionality of the document representation.
- When searching for keywords, word substrings can score as false positives, so query must match document terms.
- Additionally, a false negative scenario can happen for those documents that have a similar context but are constituted with a different vocabulary; therefore, it will not be related.
- Term order is not represented in the vector space model.

The probabilistic model is based on a series of assumptions that are made explicitly for the model to work, which frequently makes it difficult to estimate the probabilities of the relevant and non-relevant class. As mentioned in [22], "The calculation of probabilities requires the specification of assumptions that can be highly biased and inconsistent."

Nevertheless, BM25 is one of the most used models in information retrieval engines, it is especially useful when users are expected to use big text queries like complete phrases. However, with the counterpart that documents should all have a similar average length magnitude or the algorithm BM25 will score too lengthy documents higher [23]. This situation can happen in documents like books or encyclopedias.

Therefore, each model has its advantages and for this reason and as it was said at the beginning of this chapter, the selection of the model to be used for scoring documents depends on the queries the users will do, and the documents nature.

3.1.3 Indexing

Term-document matrix

The term-document matrix is an important concept to be known when studying information retrieval. However, it works as a framework for understanding posterior data structures and not as a stand-alone concept used in current practices.

For instance, let it be a concrete question for the dataset worked in this thesis, in this situation we want to analyse the emails that contain the words "Bearing", "Friction" but not the word "Turbine". The first strategy could consist on searching through the text of the documents to find all the email cases which contain "Bearing" and "Friction", flag those documents not equal and then subtract those documents in which the word "Turbine" is not present. This could be a solution fast enough to query given the computational power available nowadays, yet not the best solution for the problem as it becomes cumbersome when the document is too large.

It is a slow operation for large corpora, the logic operator NOT "Turbine" is not trivial. Scanning every word in all documents every time a user requires to find a specific term for every query is not an efficient solution. Furthermore, this becomes even more complicated in complex queries, this is because in information retrieval the ultimate goal is the idea of ranking, finding the best documents to return for query and the trivial GREP command would not suffice due to the same issues mentioned above.

To showcase a term-document matrix, below an example is shown:

Table 1: Term-document matrix for a set of 6 documents and a vocabulary of 5 words

	Email 1	Email 2	Email 3	Email 4	Email 5	Email 6
Bearing	1	1	0	0	0	1
Friction	1	1	0	1	0	0
Turbine	0	1	0	0	0	0
Short	1	0	1	1	1	1
Circuit	1	1	1	1	0	0

The words or information retrieval terms are allocated in the rows, and the columns are the documents, and what is done here is a very simple thing: for each document, fill in a cell with Boolean by whether the word appears in the document. Boolean queries like the ones before can be processed quickly using this data structure. For the query: **Bearing** AND **Friction** AND **Turbine**(Complemented), the following Incidence Vectors are obtained:

- Bearing: 110001 +
- Friction: 110100 +
- Turbine: 101111 = 100000 Corresponds to the document 1 to be the answer for our query.

The problem with this method is that it does not work once more prominent collections of data are involved.

Thus, the fundamental questions made for the design of data structures in information retrieval is how to use sparsity on the advantage of the system, and at the same time construct structures that represent data better. A proposed solution to solve those main issues and, on top of this, to create an efficient storage mechanism is by only record the positions that hold a boolean one and not the position that holds boolean zeros. Storing a vector half of it full of boolean zeros is exceptionally wasteful as mentioned in [24].

The inverted Index

The inverted index is the most relevant data structure that is present in all modern information retrieval systems. An inverted index is a data structure that takes full advantage of the sparsity of the term-document matrix and allows for fast and efficient retrieval. It is the data structure more used in information retrieval systems.

For each term, a list of all the documents that contain this term is stored, each document will be identified by a **doc ID** which is a unique document identifier or document serial number. The data structure that could be used is one with fixed-size arrays.

Table 2:The inverted index stores the ids of the documents for words occurrences

Bearing	doc1	doc15	doc35	doc43	doc64	doc114
Friction	doc1	doc15	doc36	doc42	doc43	doc135
Turbine	doc2	doc31	doc65			

In the previous table, it can be observed that "Bearing" appears in documents with doc IDs: 1,15,35,45,64,114. This is inefficient because while some words will appear in all documents, other words will appear in very few documents by the same Zipf's law, which is explained briefly afterwards.

Moreover, a dynamic index could be used where some documents will update the index if they are updated, but the main problem here will be to adjust the vector size. For this reason, variable size lists are used to build the index, and in standard information retrieval, these lists are called *posting lists*.

A variable-length array or linked list is a data structure that represents better the postings lists. Tradeoffs exists between the size of the structure and how fast the data is inserted, or in other words; the insertion speed is proportionally inverse to the size of the structure.

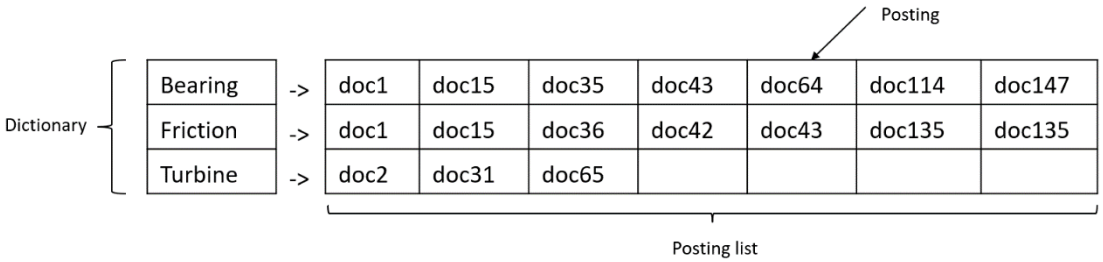

Figure 11: Representation of a posting list data structure

The terms that are in any of the documents are displayed on the left side. For each term there is a pointer to a postings list, storing the document ID in which the term occurs. Therefore a posting is the appearance of a word in a document and the collection of posting lists is called postings.

In the figure before, the terms on the left-hand side are dictionaries and on the right-hand side are the postings. An essential property of the postings is that are sorted by document ID.

There is a difference in size between the postings and the dictionaries. The reason behind this can be appreciated in the figure 11, the dictionary depends on the total vocabulary of the corpus while the postings size increase with the number of documents in the corpus.

How is an inverted index constructed? From a collection of documents to be indexed, each document is assumed to be formed by a sequence of characters.

The first stage would be to do preprocessing, this module was explained in the chapter Preprocessing, there might be various linguistic modules to modify the tokens, so a pipeline of preprocessing functions are used to transform them into a more canonical form.

In order to understand how the indexer transforms a sequence of text normalized tokens to building an inverted index, an example can be used:

Assuming two documents to index:

Doc1: The problem of the turbine was provoked by the transmission bearing.
Doc2: The turbine generator went out of synchronism due to a failure in the isolation material

The sequence of steps to go through is the following. First, the terms are sorted by primary key, putting them in alphabetical order. An alphabetical list of terms is obtained, where same terms are appearing in multiple documents.

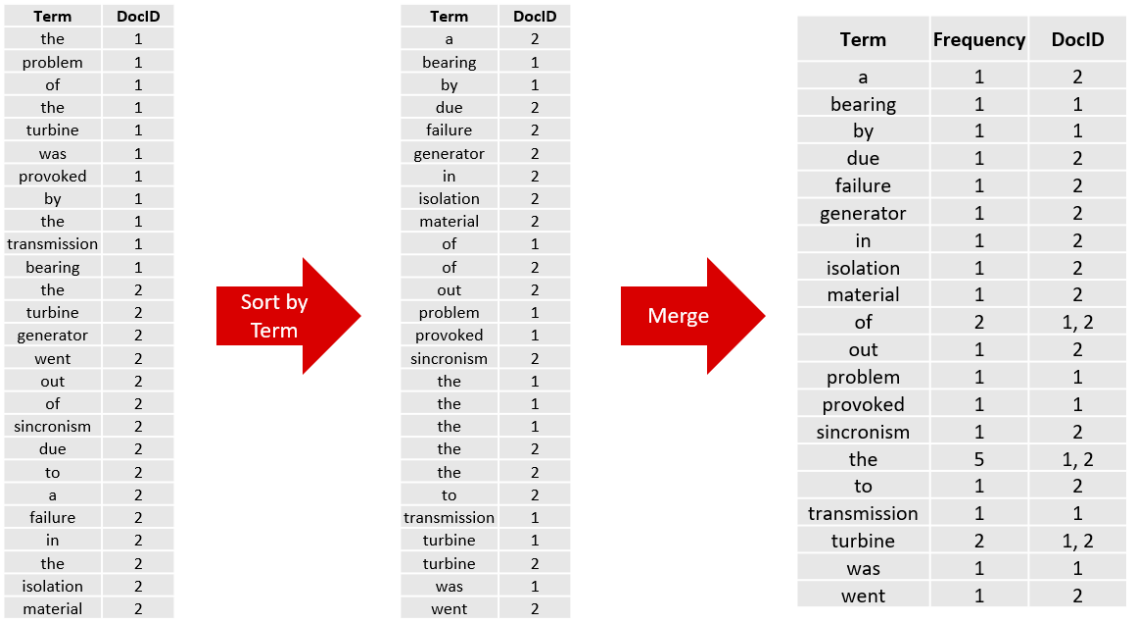

Figure 12: Representation of an index assembly

Then, a merge operation is performed to remove the duplicates and the word frequency is obtained along as the incidence per document ID.

The storage weight of this whole system is divided between the list of terms and counts, the pointers and the postings lists. In order to understand the dimensions, let us use an example:

For instance, a collection of one million documents, each with an average of thousand words. How many different terms are there? This fact is important because the number of distinct terms will tell the size of the term document incidence matrix. For this example, an estimation of 500K different terms can be done, as multiplying 500k x 1M results in a term document incidence matrix of 1500 million of boolean elements.

This is a big matrix can hardly be stored in memory. Another observation to be made is that this matrix will have **no more** than 1000 million boolean ones, if all 1000 words of the 1M documents, this deals with an extremely sparse matrix.

This phenomenon was defined under the Zipf's law, an empirical law stating that "*the frequency of any word in a given corpus is inversely proportional to its ordinal position in a frequency table*" in other words, half of the terms will only occur once [25].

Using the example explained before, there are 500.000 terms with 500.000 pointers and because the size of the collection of documents was 1 Million and the average amount of terms 1000, the postings lists is not greater than 1 billion in this particular case. Thus, the postings lists are by far the most prominent part, but they are, nevertheless, bounded by the size of the number of tokens in the collection.

Querying the inverted index

After the inverted index explanation, and how does the indexer create the mentioned index, the next question is, how should the inverted index be queried? This task is done by using a linear merge algorithm. Let the query be: "Bearing" AND "Friction". The first step is to find the documents in which "Bearing" appears:

Table 3: Bearing appearances

Bearing	->	1	15	35	43	64	114	147

The second step is to find the documents in which "Friction" appears:

Table 3: Friction appearances

Friction	->	1	15	36	42	43	135	135

Now obtain the intersection of both postings:

Table 4: Bearing AND Friction appearances

Bearing∩Friction	->	1	15	43

The procedure to obtain this intersection is by simply iterate and compare the equality of the docIDs using two pointers across the posting list of the terms. For each posting list term, starting from the beginning of the list check if the document IDs are equal or not. If they are not equal, using the smallest posting list of the two term, flag the docID in which the smallest posting list term appears and then iterate for the next term in the bigger posting list. If they are equal, flag both docIDs and iterate with the pointer.

In practice, the way documents are flagged can be done in a variety of ways, depending on the S these are defined by the scoring function that it is used. This function takes as input the query in a document and outputs different ranking scores. The ranking score is a measure of how well a doc matches the query. Scoring functions are based on ranking models, which were explained in the previous chapter.

Phrase queries

When a Google user does a query, usually this query is formulated as a phrase. The reason is that phrase queries are more practical for information retrieval system users for the ease that it provides when their query is described as a set of concepts.

For example, for the query "Azipod Turbine" the problem that first arises is the explicit and implicit meaning of the query, this happens mostly when using names. However, even though the focus is only on the explicit meaning of the query, using postings list mentioned before serve no more our purpose as it is now trivial to find a document with two words. However, it is unknown if these two words are close to each other.

To address this problem, the first attempt was bi-word indexes, which consists of indexing every consecutive pair of words as a text phrase. For the text "Overheating Azipod turbine" the following bi-words would be generated:

Table 5: Bi-word appearances

Overheating Azipod	->	9	17
Azipod turbine	->	7	9

This task shall be done for all pairs of words in our text corpora, generating an index of bi-words and solving two-word querying. Furthermore, longer phrases can be queried by doing the intersection of the postings of this bigrams, so for the query "High Voltage Insulated Transformer", the bi-word query would be:

"High Voltage⋂Voltage Insulated⋂Insulated Transformer"

Still, with this query methodology, the main problem that arises is the dramatic increase of the index size as a result of an increase in the size of the dictionary. For this reason, anything more significant like tri-words becomes overwhelming to handle, and still, there is a small space to get false positives if there is no linear processing of the documents.

The next solution to overcome the problems mentioned before is a different type of index, one who has information of the position of each term, for each document the term appears. This is called the proximity index or positional index:

Table 6: Example of a positional index

Bearing	→	1,3	2,1	3,3	4,3
Friction	→	1,1	4,2	5,8	
Turbine	→	4,7	6,1		
Overheat	→	5,4	6,3		

For the previous data structure, the word "Bearing" appears in the corpus four times, it does not appear in doc5 and doc6 but appears, in position number 1 in position 3, in doc2 in position 1, in doc 4 in position three and doc 3 in position 3, for instance.

This positional index makes it possible to phrase query it by using a two-level linear merge algorithm, first intersect the documents ID's and secondly check if there are compatible positions for the words occurring in a phrase. For instance, it can check the distance between the position of two words:

$$pos(Friction) - pos(Bearing) = 1. \quad (20)$$

Table 7: Positional indexes

Bearing	→	1,2	2,1	3,3
Friction	→	1,1	4,2	5,8

This operator can be altered to check for how close two terms are in a document, this "near" operator makes room for a variety of methods to query the index. For instance:

$$|pos(Friction) - pos(Bearing)| < 3 \quad (21)$$

This equation is checking how close are the terms Friction and Bearing, and as the order does not matter, the absolute value is being evaluated. If the order would matter, the absolute value does not need to be checked.

An additional feature from Text-documents is meta-data like title, author, date. Also, the text has a structure like a chapter, section, paragraph. With the proximity index, there is room to store this information as tags.

How shall this information be stored? The cheap approach would be to create a separate index for each field, in a SQL style, but this does not work out well as the data is being fragmented. Another

approach is to push the structure of the text into the index values, so for example introduce a token that would be "Machine: Turbine" or "Machine: Generator". But the most common approach is to construct an Extent Index

Extent index

An Extent index amplifies the versatility of the proximity index by adding tags as additional terms:

Table 8: Example of an extent index

Bearing	→	1,3	2,1	4,3	3,3
Friction	→	1,1	4,2	5,8	
Transformer	→	4,7	6,1		
Overheat	→	5,4	6,3		
Generator	→	2,7	3,5		
Electrical Machine	→	2,7	3,5	4,7	6,1

These representations allow the overlapping of multiple terms, so there would be positions in the text where there would be two terms simultaneously and also allows the spans of a region of text, words in the span belong to the specific field. This span can be stored as with the format 2,1:2

Index compression

Indexes are undoubtedly significant, even though storing them is not a problem, it is costly to bring the whole index into memory. For this reason, the strategy is to bring small pieces of the index into memory to process it. To do this task efficiently, index compression is performed to reduce storage space and improve Input/ Output time. The indexes explained in the previous sections are formed by extensive amounts of numbers. With index compression, large numbers are converted to small numbers, and afterwards, these small number will be represented by the least amount of bits possible.

An inverted list is a sorted list of documents ids, as the document count is typically big, it takes a decent amount of bytes. So for instance, to store 10 thousand million document ids, we would need 8 bytes integers to represent and store this document ids.

The basic idea of delta encoding is, while the numbers themselves are significant, the difference between the numbers of the document's ids are small, precisely because this number sorted the index. By using the deltas of the id numbers and sequence them together, it is possible to replace the entire index with the deltas.

This encoding does not work well for infrequent words, but it does work well for frequent words. So, for terms that tend to appear all over the corpus, the occurrence between subsequent documents will be much higher, which leads to a smaller difference between document id numbers.

Query execution

In the previous section, it has been explained the different rationales that made room to different index strategies. It has been explained how to form up this index and basic ways to compress it. Additionally, some basic ideas on how to query the index were introduced, this section contains an in-depth analysis and explanation on how an index is queried, the different methods and their advantages and drop-backs:

Index query execution main goal is to output results most quickly and efficiently possible. Query execution depends on the score function used. If the score function is set-based, e.g. a boolean AND, the documents can either match or not a query. While if they are rank-based like TFIDF weight

scores, most documents will match, but scoring function will compute a relevance to rank how close a document is to a query.

The two strategies followed for query execution are:

Doc-at-a-time execution

Assuming an inverted index and an entry query "Overheating Transformer Short-circuit" and the scoring function the weighted sum of those three terms with the form:

3* # of docs (Fault) + 9* # of docs (Transformer) + 3* # of docs (Overheat)

The basic algorithm to do this is the merge algorithm, using a pointer start comparing the different postings comparing K different postings lists:

Table 9: Example of postings list comparison

Overheat	→	3:1	4:1	5:1	X2
Transformer	→	4:1	5:1		X5
Fault	→	3:1			X9

The scores obtained are:

Doc3 Score: $2 * 1 + 9 * 1 = 11$

Doc4 Score: $2 * 1 + 5 * 1 = 7$

Doc5 Score: $2 * 1 + 5 * 1 = 7$

Term-at-a-time

Tries to flip the problem on the other side, so instead of trying to merge lists, what is done is incrementally compute the scores of all the documents updating them one term at a time.

For this purpose, an array with a slot for each of the documents is retrieved. Then fetch the inverted list for Bearing, so it occurs once in doc4, once in doc5 and Bearing weights two in the query so what is multiply two by the frequency in each document and update the score in the array I just initialize. The next word is Overheat, which occurs in doc3, doc4 and doc 5, one in each doc. So as overheat has the weight one, the array is updated accordingly. Now for Transformer, it occurs in doc3, but the weight is ten, so updating the documents we obtain the result. [24]So what has been done is taking the term and updating all the documents, term by term, so we are computing the scores in no particular order, which is an important part when talking about optimization, we can also process the terms in any given order.

Doc-at-a-time only emits the scores when they are not zero; it only emits when the scoring function yields a non zero value. Term-at-a-time computes the scores for all documents, so the final task when using this approach is to extract the non-zero values from the final scores array and usually sorting them or ranking them by score. The complexity is linear $O(n + N)$ with n: total length of all lists and N: number of docs in database. Usually, $n >> N$ but it can happen that $n << N$.

3.2 Information retrieval engines

3.2.1 Technologies research

The primary objective of the thesis is to provide knowledge accessibility to customer service information tools. For this reason, building an information retrieval engine was the primary idea

selected. The wide variety of solutions in the market needed a more reasonable approach. For this reason, a set of different requirements were empirically researched to narrow down and select the most optimal technologies that could suffice for building the tool. Hence, the solution must:

- Be able to provide open source full-stack information retrieval solutions.
- Be able to manage nested data structures.
- Be able to provide fast and relevant searches.
- Be able to integrate easily with cloud services from stakeholders.
- Be able to scale horizontally easily.
- Be relevant enough to find online support to solve problems.

In order to investigate and study the different technologies, the study did consider different documents mentioned along with this chapter but also the study done in db-engines.com. This website compares different databases and search engine by calculating a score based on a series of parameters and rank them along time. Therefore a general relevance across time can be evaluated to help understand the interested in these technologies how they have been evolving. The scoring calculation is based on six parameters; they can be found in [26]:

- Number of results appearances in search engines queries, obtained from Google, Bing and Yandex
- Frequency of searches as a general interest indicator, obtained from Google trends.
- Frequency of technical discussion, as an indicator of Online support the system has obtained through Stack Overflow website
- Frequency of mentions in job search portals, as an indicator of the relevance of the system in the job market, obtained through Simply Hired and Indeed websites.
- Frequency of appearance in professional online profiles, as an indicator of relevance in the professional market, obtained from Linkedin and Upwork.
- Relevancy in Social networks, obtained by the counts of tweets the system is mentioned

July 2020

Rank			DBMS	Database Model	Score		
Jul 2020	Jun 2020	Jul 2019			Jul 2020	Jun 2020	Jul 2019
1.	1.	1.	Elasticsearch	Search engine, Multi-model	151.59	+1.90	+2.77
2.	2.	2.	Splunk	Search engine	88.27	+0.19	+2.78
3.	3.	3.	Solr	Search engine	51.64	+0.38	-8.00
4.	4.	4.	MarkLogic	Multi-model	11.67	+0.43	-2.14
5.	↑ 6.	↑ 6.	Microsoft Azure Search	Search engine	6.55	+0.25	+0.13
6.	↓ 5.	↓ 5.	Sphinx	Search engine	6.54	+0.19	-0.15

Figure 13: Comparison of top-6 relevant search engine technologies

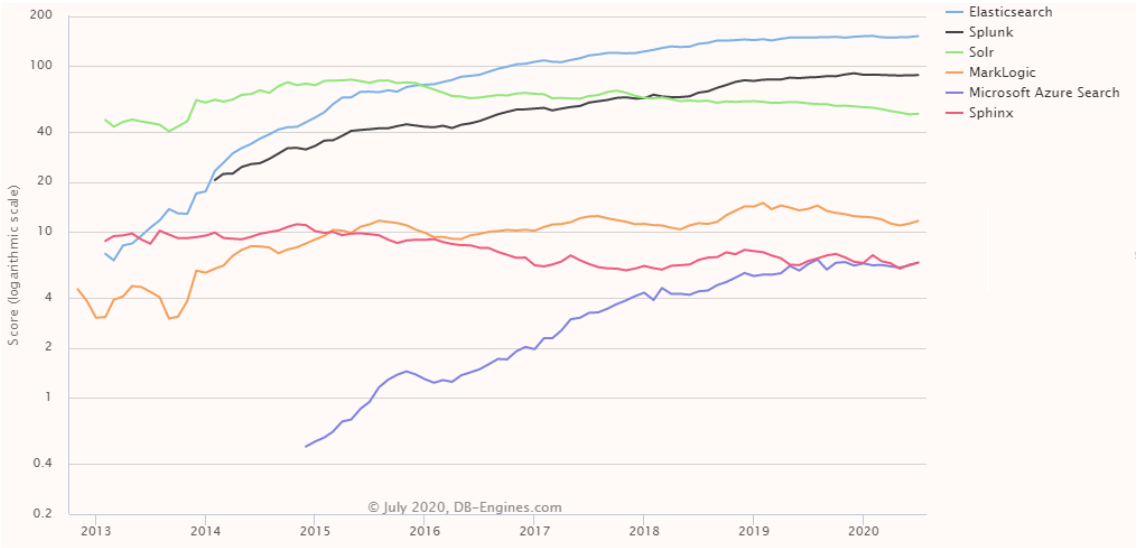

Figure 14: Comparison across time of the different search engine technologies

In the two previous figures, it can be observed the top 6 search engines, ranked by db-engines. Elasticsearch and Solr are the most relevant full-text search engines. Splunk has become very popular in the last years, [27] [18]. Even though Solr popularity has been declining since mid-2017, the technology still is a good alternative for full-text search. Marklogic, Microsoft Azure Search and Sphinx will be out of the scope of this study due to many reasons:

- Both Marklogic and MS Azure Search are commercial solutions, making their code inaccessible to the investigation.
- Their low popularity in comparison with Elasticsearch and Solr, that causes them to have a community less active, hence having less online support.
- Sphinx does not provide good Big data functionalities, replication methods and MapReduce. Additionally, it does not provide consistency for a distributed system.

According to (Nikola et al.,2016) [28], both Elasticsearch and Solr have very similar functionalities. Additionally, they are open source, but while Solr is community-based under the Apache project, Elasticsearch code is managed by Elastic company. This point is essential to understand how Elasticsearch has evolved, as a commercialized product, which offers its functionalities for free but at the same time, offers the possibility to mount the IR system in their cloud.

Elasticsearch success resides in its product offer, providing a big variety of technologies to manage your information retrieval system completely. In [29], Jonathan Blasenak states that while both Apache Solr and Elasticsearch are based on Apache Lucene, their implementations approaches have differentiated them.

While Apache Solr is highly customizable to fit a variety of IR needs, Elasticsearch provide a more out-of-the-box solution. Elasticsearch shall be used if there is a specific short-time frame to create a minimum valuable product, as it has been designed to be easy to install and start working. On the other hand, Apache Solr can be more valuable if the company have already existing resources built to work with Solr, e.g. any Apache software like Hadoop .

If there is a particular interest in Data visualization, Elasticsearch provides Kibana, a database visualization tool that connects to Elasticsearch clusters to provide fast and easy data visualization and cluster management.

In terms of security, both technologies can have custom plugins developed, but to be more specific, analysing the services provided:

Solr provides a basic Authentication service through Zookeeper, Hadoop and Kerberos, but again depends on the company framework to be working under the Apache framework.

On the other hand, Elasticsearch provides a Native support through X-Pack plugin, additionally support for different security protocols (e.g. LDAP, Security Assertion Markup Language and Public Key Infrastructure).

Analysing the data structures compatible. Elasticsearch is particularly good at supporting deep nesting structures. On the other hand, Apache Solr, has the notion of parent-child but are indexed separately in two different indexes. This makes it especially limited when aggregation operations need to be performed.

In terms of Scalability, Elasticsearch provides near real-time, scalable service. Elasticsearch architecture makes it easy to scale up horizontally from the MVP. This architecture is divided in clusters or shards, and it provides automatic node management in terms of rebalancing after node addition or data reorganization after a node failure. This makes Elasticsearch exceptionally resilient and fault-tolerant. Solr has more problems on regards scalability, "Scaling requires manual intervention for shard rebalancing" as mentioned in [29].

Analysing Big data support and cloud computing: All major cloud providers (Azure, AWS, Google cloud) offer fully managed solutions for Elasticsearch. Additionally, Elasticsearch provides ES-Hadoop for MapReduce operations. Additionally, Elastic offers Elastic Cloud, which the Elastic hosted cloud solution and by purchasing a subscription, offers the possibility to host a pipeline in any of the major cloud provider mentioned before, with the value-added of providing with a installation environment that can get set up full Elastic stack ELK working in very little time.

For all of the above mentioned, Elasticsearch was chosen to be used for this project. On the next sections can be found a description of how Elasticsearch works internally, the basic configuration ES provides as well as an explanation of the data scheme and API exposed.

3.2.2 Elasticsearch

Elasticsearch is an open-source information retrieval service based on Apache Lucene. It exposes a RESTful API that makes it easy to interact and configure. It provides a distributed full-text search with the possibility for multitenancy. Additionally, Elasticsearch inside architecture is built to provide easy horizontal scalability. Examples of companies using Elasticsearch are Stack Overflow [30], Netflix [31], SoundCloud [32] which serves as a prove of concept for the potential of this technology.

Full text search is not the only functionality Elasticsearch can do, it is possible to write queries that aggregate data and use the results for making charts or even filters for your search engine. Even though Elasticsearch is not a business intelligence solution, it is possible to obtain valuable information out of the data stored within it. An example would be to store data from different IoT devices like proposed in (Marcin Bajer,2019) to be monitored thanks to Elasticsearch GUI package called Kibana. [33]

3.2.3 Inside Elasticsearch: Apache Lucene

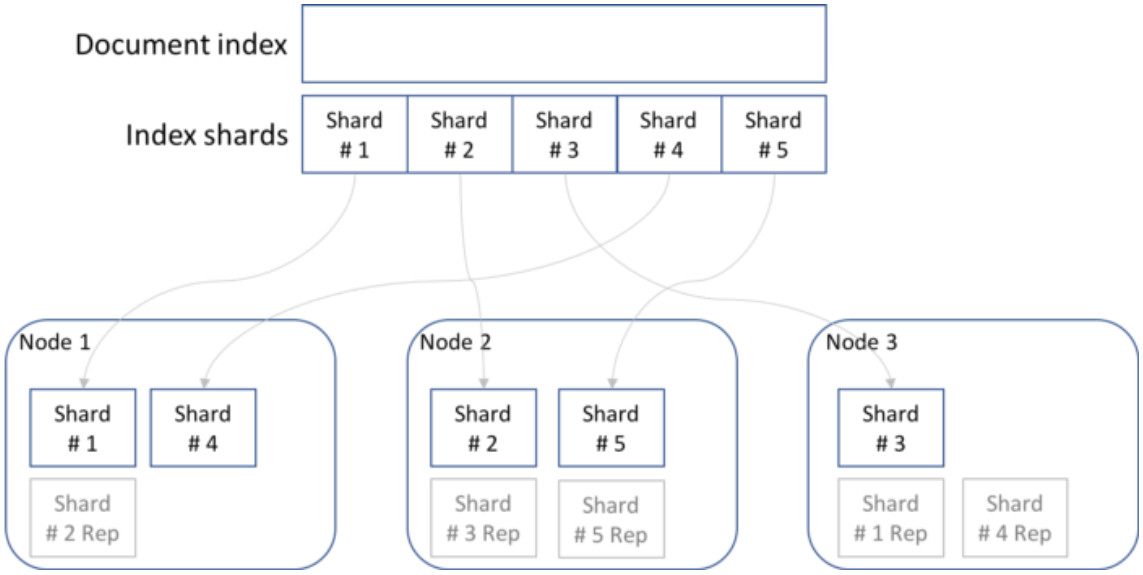

Figure 15: Elasticsearch cluster formed by three nodes with an index sharded in 5.

An Elasticsearch cluster works by setting up nodes. Each node is formed by different shards. These shards are partitions of the data indexes and can be understood as self-contained Lucene indexes. These Lucene indexes are formed by different Lucene segments. Lucene segments are formed by certain data structures. These data structures depend on what type of data they store. Examples of this type of data structures are inversed index, stored fields or document values [34].

When Lucene searches across an index, it searches all the segments and merges the results. **Segments** are immutable; this means they never change. If a segment is deleted, it is not removed from the segment but flagged, so the merge algorithm ignores it. When an index is updated, the segments flagged are deleted, and then the index is updated.

Segments are created by Elasticsearch every refreshing interval, which by default is 1 second, in the meanwhile, Elasticsearch buffers all the information until the next updated is executed. Once the information is updated to the index, the new data will be available for a search. Therefore having more documents can lead to smaller indexes as a new update can cause more data compaction.

This whole process occurs within a Lucene index, which is a shard in the Elasticsearch index, which is allocated across nodes in your clusters. When searching across shard is the same as searching across segments. Shards are used to evenly distributed data across one index.

In this manner, write requests are routed to the primary shard, then replicated while read requests are routed to the primary or any replica shard. This archetype makes read requests much faster as the request can spread out the load across all shards efficiently. Hence, the more replicas in the cluster, the more reading capacity will be available with the counterpart of needing more infrastructure to store these replicas.

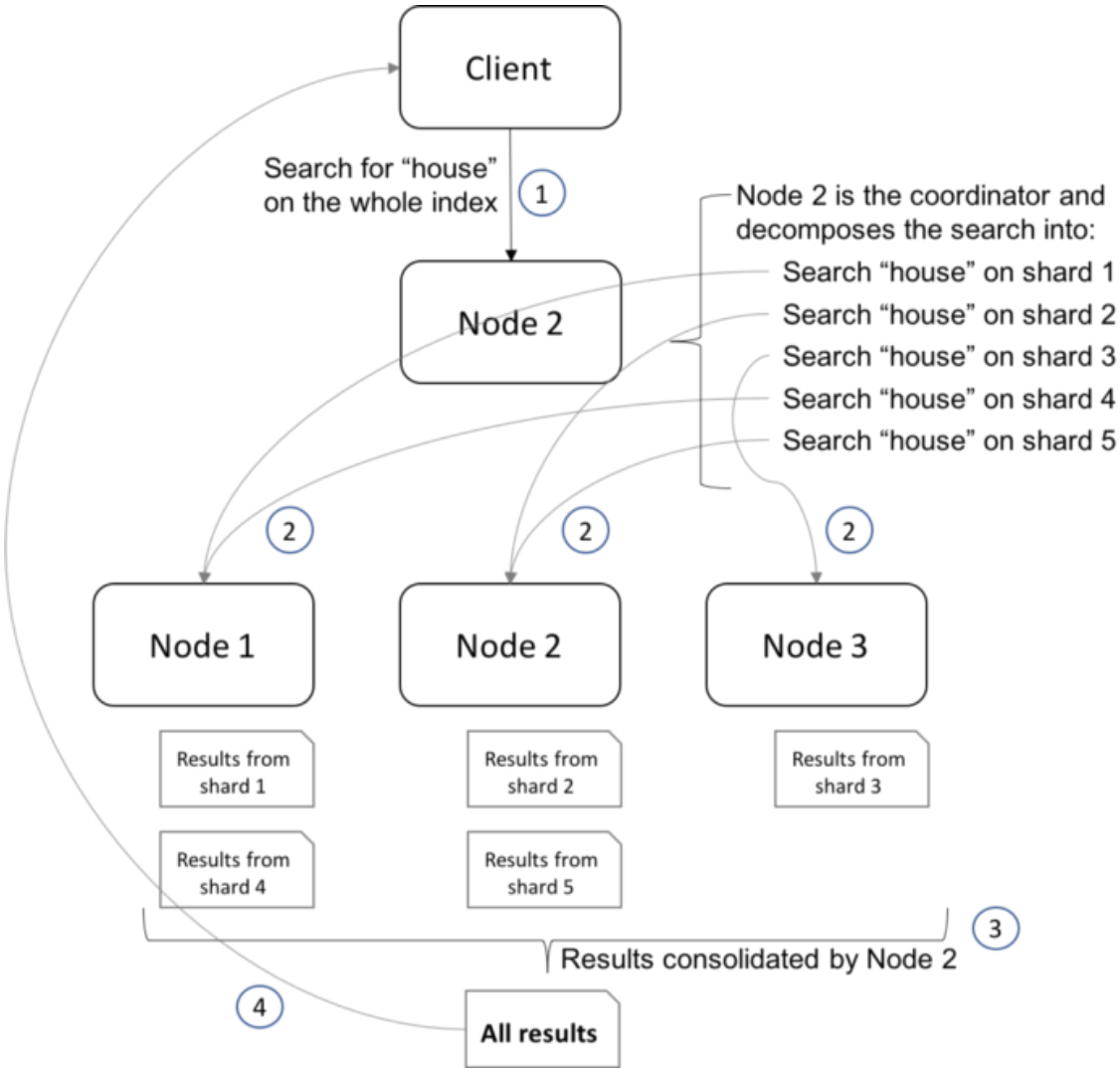

Figure 16: Order of the execution in which a query is processed

On the other hand, shards need to be defined when the index is created; hence, if a system starts getting overloaded, there is no way to turn one shard into two. For this reason, planification is remarkably essential. Another standard error is to create many shards to then dynamically split them, but this will be counterproductive as Elasticsearch will need to search over multiple instances of duplicated data [35]. It is imperative to find a balance in shards count for this reason.

Taking a look at the figure 15, the elastic cluster has three nodes setup with five primary shards and fice replicas. Assuming node 1 fails, primary shard 1 and 4 and replica shard 2 will be lost until recovery, but because there is a replica of shard 1 and 4 available in node 3 the service can up. This system architecture is the backbone of Elasticsearch, creates a that is very fault-tolerant as requests will be hitting the nodes on a "round-robin" fashion. [34]

Types of nodes

Elasticsearch has different types of nodes, this node types achieve the task of different roles in the system. Having different node roles provide some advantages in big data cloud services. In Elasticsearch, the following node types can be configurated:

- Master Node: Manages the Elasticsearch cluster, which means it assigns shards to different nodes, being able to create of delete indexes across all clusters. Can only work on 1 cluster at a time while updating the state changes produced to other nodes. These nodes will send

an affirmative response to the master node (). It is recommended to have three master nodes, 1 dedicated and 2 assignable for high availability. Having a master node only focused on cluster management its recommended for high availability

- (Talk about how master nodes do the votation to select)
- Data node: store the inverted indexes and other types of data. This is the default node type in ElasticSearch
- Client node: Load balancing node, evens the workload of different nodes working

Basic configuration of Elasticsearch

To configure Elasticsearch, basic settings need to be applied even though Elasticsearch predetermined settings are enough for the objective of this project, a deep understanding of this settings can teach the reader more about how the architecture works.

Path settings: To define where the data and logs will be store. Since the MVP will not be upgraded is not that important. However, for production mode, it will be essential to set a custom path for both logs and data to avoid the risk of data deletion when the Elasticsearch is updated to the latest version.

Cluster name: If different nodes to join a cluster, they all need to share the same cluster.name. It is recommended to use a name that is unique and describes the cluster purpose at some sort. Using the same cluster name in different environments can lead to nodes connecting to different clusters.

Node name: Is used as a readable identifier for an instance of Elasticsearch. This will be visible in the response of many APIs

Network Settings: Elasticsearch binds to localhost URLs, so in this way it is possible to try a single development node on a server. Even though there are various network settings, in order to create an MVP, a loopback address is enough. An important note to take into account is that as soon as a custom network is provided, Elasticsearch will start performing several start checks as the system assumes the proceeding from development to production mode.

Discovery settings: These settings will setup how nodes in the cluster can discover each other and elect a master node.

Heap Size: Is the amount of memory that the Java Virtual Machine that is running inside Elasticsearch has available. Usually, the heap size is set up to no more than the 50% of the physical RAM available, as Elasticsearch needs memory for other tasks regarding communications and efficiency of the whole system.

Elasticsearch index mappings

Elasticsearch has a free scheme database or No-SQL database. This fact means that Elasticsearch stores information in the form of documents. Documents are an information type which its unit consists of a file with a concatenation of fields followed by values. These objects have the following form:

```
{


  "name": "StayFocus Extension",
  "description": "Enhance your working experience!",
  "version":"1.0",
  "manifest_version": 2,
  "browser_action":{
    "default_icon": "img/app/logo.png",
    "default_popup": "popup.html",
    "default_title": "StayFocus buddy"

  },

  "permissions":["tabs",
    "storage",
    "activeTab",
    "cookies",
    "http://*/"
  ]
}
```

Figure 17: Example of JSON nested document

In figure 17, it can be observed different fields: "name", "description" and their values. The advantage of using this type of data structure is that it allows the use of nested information. This is, instead of storing a value, store another document as a value. Looking again at the figure, the field "browser_action" contains a small document with three fields: "default_icon", "default_popup" and "default_title".

Now that JSON document data structures are explained, Elasticsearch mappings will be more straightforward for the reader to understand.

In Elasticsearch, setting up a mapping serves the purpose of:

Setup the scheme of the data architecture that will be input

By setting up the data architecture previously, we can prepare the system to reject the ingestion of data with different fields that the ones we are interested in. In this way, the system is protected, and all the database stays uniform

Format precisely the data types for each field.

Elasticsearch provides a wide variety of data types to set up the fields in the mappings: *double, date, long, text, keywords, Boolean or IP*. It is also possible to set the hierarchical nature of the fields with *object, nested.*

Setup custom rules to control dynamic mappings.

In case that it would be of interest to inject data which is not present in the mappings, it is possible to set rules for the types of data that will be accepted and also to format this data in a specific way. For example, imagine the case where a second date field is going to be injected in the database. But this date field needs to have the following format MM/DD/YYYY instead of DD/MM/YYYY to

have the same format as the first date field previously present in the database. Thanks to this configuration it is possible to create a rule to reformat it dynamically.

Setup control parameters to avoid mappings explosion and out of memory errors.

It is possible to set limit parameters to:

- Set a limit to the total number of fields in the index.
- Set the limit of the depth of and field; this is the depth of the nested fields
- Set the limit of the length of a field name

Setup indexing conditions to specific fields.

It is possible to set a condition to text fields to avoid their indexing according to their length.

Setup specific scoring algorithms

Different scoring functions can be set up in Elasticsearch for different fields. For example, instead of using the Okapi BM25 algorithm that Elasticsearch uses by default, the tf/idf algorithm can be used to rank specific fields from a database. This functionality is most useful for text fields when performing full-text searched, but cannot apply to other field types like keywords or dates.

In this chapter, necessary information of Elasticsearch configuration has been explained. With this information, the author aims to provide the reader with general knowledge to understand the implementation better.

Chapter 4

4. Implementation

4.1 System architecture

In the previous chapter, it has been explained the technology that will be used as an information retrieval engine. Nevertheless, this is only a component of the implementation of the full-stack solution needed for a minimum-valuable search engine. To create this solution, system architecture was designed to power the technology is chosen.

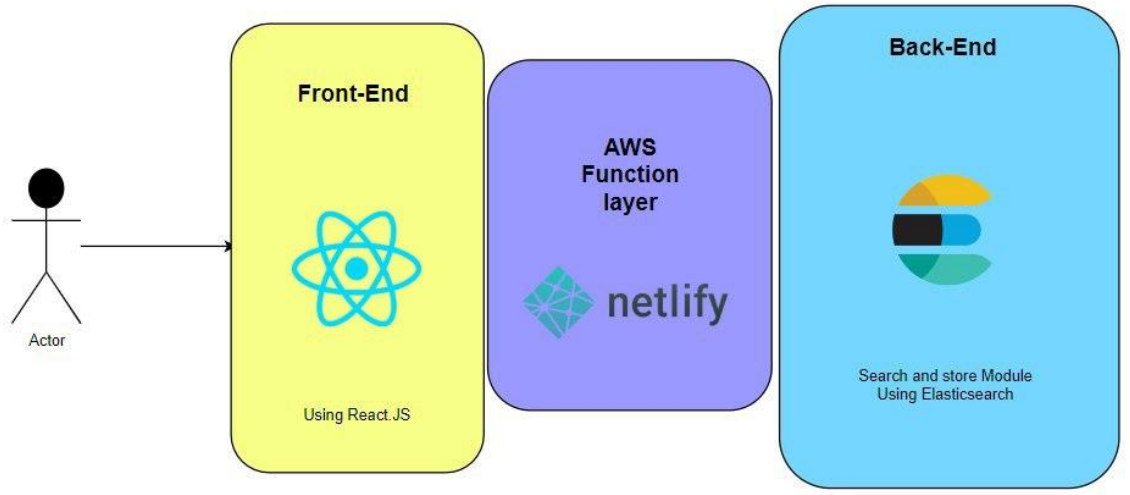

Figure 18: Representation of the full implementation.

- A static web service, implemented in React.js, using a the SearchUI library provided by Elasticsearch, deployed using a Server-less lambda function by Netlify
- A "dockerized" instance of Elasticsearch as the back-end, exposed with an API.

4.1.1 Dockerized instance of Elasticsearch

Elasticsearch provides a highly effective way to create a cluster using Docker. Docker is an operative system for containers. In a similar way that a virtual machine virtualizes a server's hardware, containers virtualize a server's operative system. Containers provide a standard solution to develop applications, transfer the code to a production environment quickly and provides an efficient way to monitor the application's resources to maximize its efficiency [36].

Additionally, to Docker, Docker-Compose was used. Docker-compose is a software tool that helps to create containers clusters in a very efficient way by reading a text file in YAML format, previously created with the configuration of the clusters wanted.

When a cluster is created, the backend exposes an API, which can be used to interact with the configuration, indexes, plugins and of course, send a search query.

As it will be explained in the Elasticsearch backend chapter, the API gives control to many different functionalities of the Elasticsearch, from which some of them should have limited access like the data deletion.

4.1.2 Static search portal

As it will be explained in the upcoming chapter front-end, the search portal was developed using a SearchUI template provided by Elastic, the company behind Elasticsearch. The search for a front-end component was not as extensive as the for the back-end. The reason is that a functional solution enough for this thesis. Nevertheless, as Elastic provides an out-of-the-box front-end template solution that harmonizes perfectly with their technology, this was chosen. However, its election was not only because of the functional requirements were extensively fulfil, but also because it offers other features not needed but yet good that will benefit the front-end. Like a minimalistic design, a serverless deployment and static website architecture that are extremely interesting to learn and familiarize as they are part of the state-of-the-art features in software development.

This library provides a headless or static website architecture, which can be thought of as a web architecture with empty components. The content of these components is loaded via API requests. These API requests are made, in our case, against the Elasticsearch API endpoints.

A significant benefit of static web services is that it works as a small program or function. This fact opens the possibility in software architecture to deploy in a specific cloud-service called FaaS or Function-as-a-Service. This means that by deploying our small program or function into a 3rd party cloud service, the service provider will take care of the services that regular server-hosted websites need to take into account, like load balancing, monitoring, security patching, logging [37].

FaaS providers can replicate the code deployed to satisfy the demand for web services. And most important, it works proxy request for the back-end, limiting the access of malicious requests, in this our case, to the Elasticsearch API

The following figure gives a good understanding of FaaS.

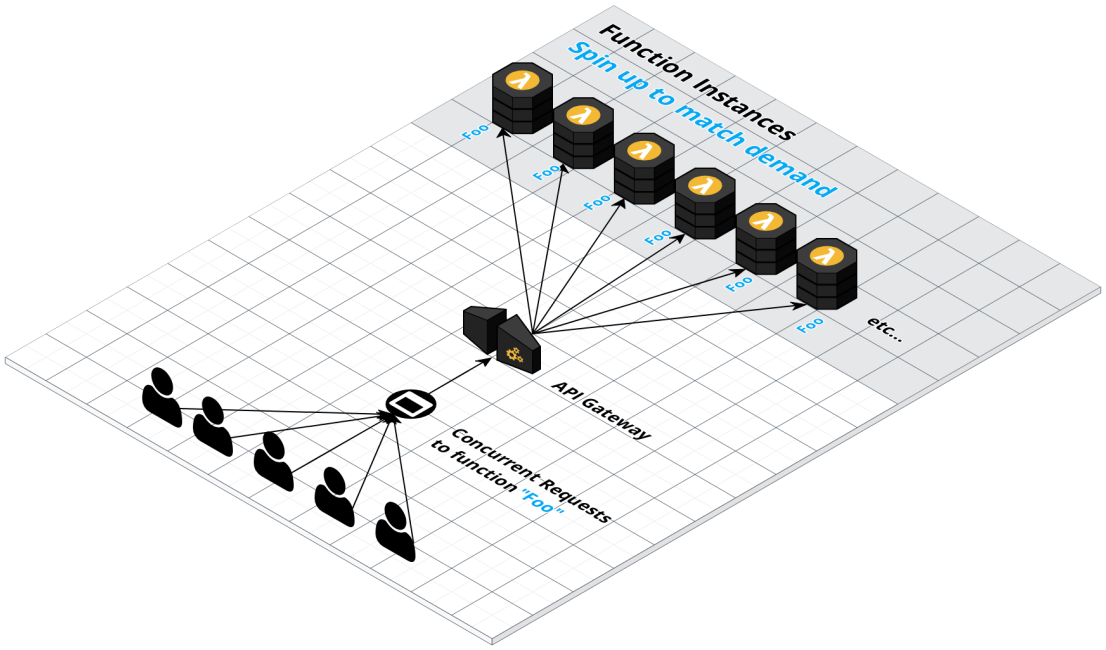

Figure 19: Function instances replicate on-demand to concurrent connections

This circumstance showed to be a promising step to take for the MVP for the following reasons:

- The most important, add a first security layer to the system as it proxies the requests to the API backend.

- It provides a ready-for-deployment solution

The FaaS provider used is Netlify. Netlify is a technology based on Amazon Web Service Lambda Function that makes the process of deploying AWS reasonably easy. Netlify provides the functionalities of AWS Lambda Functions without needing to have an AWS account.

Before explaining the Elasticsearch backend, a description of the data structures used in this thesis is explained. Afterwards, a description of the Elasticsearch configurations follows. Subsequently, an explanation of the front-end functionalities and implementation is provided. Finally, a description of the process to discover the best query strategy to configure the project is shown.

4.2 Dataset

For the creation of this MVP, an email dataset was provided. This dataset consists of two SQL tables which have a parent-child relationship between them. The parent table is formed by Cases, each case has 51 different features, one of it being the case ID which is the unique identifier of these cases.

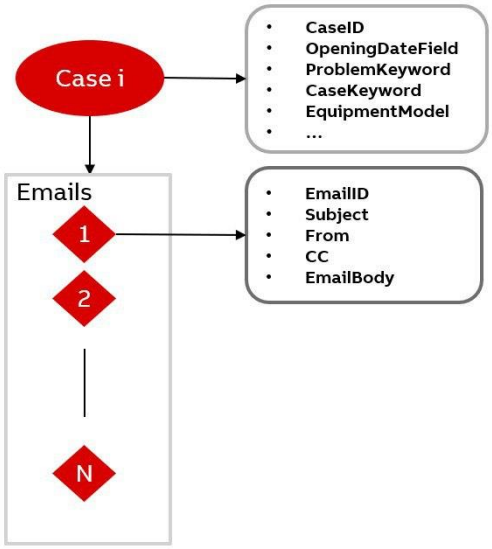

Figure 20: Case-Email representation with their metadata.

Each case has linked set of documents corresponding to emails conversations exchanged by the customer and the customer support unit. The emails data have standard email metadata: case ID, unique email ID, subject, title, sender, receiver, Cc, Bcc and the most relevant one, Body.

An important fact about this email Body is that this field does not contain individual email conversations but a chain of conversations from different parties. This will be a decisive impediment that limited the query strategy, and it is explained in chapter 4.5 Refining the query strategy.

The database documents consist of 8465 cases and 141203 emails, according to the database source information. For exploration purposes, the data was downloaded locally to perform fundamental data quality analysis.

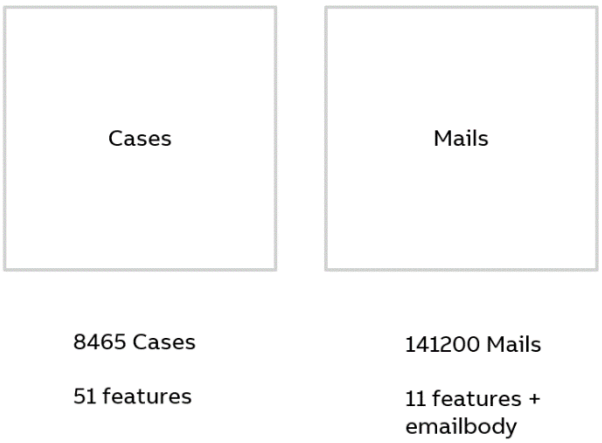

Figure 21: Representation of the database tables

The data quality in Case IDs was not optimal. Basic string pre-processing tasks needed to be done on the data. As all the "Caseid" have a standard form, a regular expressions algorithm was used to filter all the noise in the string. The same method was used for any other keyword data type in the database. The Keyword datatype name refers to any text data or string data that is not free text.

After obtaining the same count of Cases and Emails as the database source, the other metadata keyword cases were cleansed from similar string noise. As a significant number of this case metadata has a free text format, the data was cleansed after different trial, and error cycles were the most common type of noise were new lines, space tags "\r,\n,\r\n".

Apart from the noise that different data fields had, the main problems found in the database where the discovery of cases with no emails and emails with no parents. From now own, the first ones will be referring to as childless cases and orphan emails.

4.2.1 Childless cases and orphan emails

After the cleaning tasks were finished, the next step was to find the relation between cases and emails, this is, to map the exact number of email by case to create the data structure but also to find out how many emails have each case. It was found out that there were a significant number of orphan emails.

As it was discovered in the Contextual Inquiry explained in chapter 2.3.1, the value for the users in this dataset resides in the relationship between case and email. The case features are needed to bring a notion of what the customer issue consists in: e.g. the fault code, date of the incident and other relevant information while the email body helps the users figure out the actions that need to be taken for their new case in hand. Therefore, it was needed to find the join between both tables and separate those cases and emails which were not linked for future exploration.

Figure 22: Representation of the union of both tables, showing part of the data non-linked

Therefore, 38512 emails were found not relevant and needed to be separated from the dataset. Also, there were found 230 cases with no emails, while these cases have metadata attributed to them, it seemed there were no emails bounded to them. This problem raised the question: if there was any relationship between the group of emails and cases non-linked, how we could find out and what techniques can help?. So, the question raised on how this could be achieved, as this study would be out of the scope of this master thesis, an overview is given in the Future Steps sections on how it could be achieved.

4.2.2 Blank data fields

Another problem present in the dataset was blank data fields. This problem was still present after the deletion of non-linked cases and emails.

As learned from the personal inquiry mentioned in chapter 2.3, since the feature fields are the primary source of information to profile cases into different categories, it became a problem that needed more investigation.

After consulting with the customer support managers, the reason behind this problem is that the unit adopted a new framework through which more features started to be used to define and classify incoming cases. Additionally, after this framework started to be used, the unit focused more on correctly inputting the features from the cases.

Data analysis was performed to understand if there had been any trends in the cases blank fields input in time to confirm the points mentioned in the previous paragraph. The database has cases that dated from 2013 until 2018. The number of cases per year is the same with a variation of ±5%. The first year, 2013 the number of blank fields divided by the total fields was more significant than 50%, from 2014 and so on, fields left blank decreases through time. After consulting with the customer support unit, this was due to a change in the software used for customer support and data input habits promoted by the managers of the unit.

To give a better view of the blank data problem, the percentage of fields in blank in the database was 42% this is, from 403k data entries, 167k were blank. This amount of blank values needed to be addressed before ingesting it into Elasticsearch as the database have 48 features, the survey mentioned in chapter 2.2 help to understand that the customer support engineers observed not all the features. The CS engineer valued as relevant less than twenty of the features from these cases. This situation is because they are used to go through extensive amounts of meaningless data, so as a result of their

acquired expertise through the years has consisted on understanding the data that help them find an old case close to the one at hand easier.

In the table below, it can be observed a presentation of the features more valuable for the users. The names of the features have been changed to ensure data privacy; nevertheless, the new names given try to provide a standard view of the meaning of each feature.

To be in the table, the features needed to be scored on average, a score mark greater or equal to 6. These 19 features were the resulting output. This table will have a significant impact on the querying and the scoring of the documents. This statement will be addressed in chapter 4.5, Refining the query strategy.

Table 10: Data users in the project, along with the parameters that decided their election.

FeatureNames	DataType	AvgOfUsefulness	Blank Values
OpeningDateField	Date	6.5	0
DeliveryDateField	Date	8.5	1310
CaseId	Keyword	10	0
ProblemKeyword2	Keyword	8	0
CaseKeyword	Keyword	9	9
ProblemKeyword1	Keyword	9	9
EquipmentModel	Keyword	9.5	268
OrderID	Keyword	10	618
EquipmentID	Keyword	10	815
EquipmentPower	Integer	6	1312
EquipmentVoltage	Integer	6	1313
EquipmentVelocity	Integer	6	1315
EquipmentFrecuency	Keyword	7	1321
EquipmentDisipation System	Keyword	6.5	1330
EquipmentMounting	Keyword	7	1331
CaseName	Text	8	703
Contact	Text	6	1060
ShortDescription	Text	10	1550
		Total cases	8231
		Total blank cells	14264
		Total cells	148158
		Percentage of blank fields	**10%**

4.3 Back-End: Elasticsearch cluster

As mentioned in chapter 3.2.3, Elasticsearch was chosen as the back-end of the service. This chapter coverts the setup and configuration of the Elasticsearch cluster.

As mentioned in at the beginning of chapter 4.1 System architecture, the Elasticsearch cluster was containerized in a virtual container using Docker.

By creating a docker-compose YAML file, Docker Compose reads the file with the instance configuration, downloads the image, and creates the virtual container. In Annex 3, the docker-compose YAML with the configuration used for the Elasticsearch setup can be found.

Due to the small database size, less than 1 GB, the Elasticsearch cluster was created with one node and only one shard. As mentioned by Dahlqvhist in [38], "*There is no fixed limit on how large shards can be, but a shard size of 50GB is often quoted as a limit that has been seen to work for a variety of use-cases.*", so one node with one index shard is more than enough for the MVP development.

Once the virtual container is operating normally, the first step to be made is to create an index with a mapping in JSON format. The mapping wrote to create the first index is shown in Annex 1.

To create the index in Elasticsearch, a POST HTTPS request to the localhost endpoint needs to be made. For this step, Postman is used as it makes it easy to make and store different API requests. In the following figure, it can be observed the POST request made with the response from the server confirming the index was created.

Once the Elasticsearch cluster was set up and the index created, a pipeline was created to extract, transform, and load the data into Elasticsearch. The main tasks of this pipeline are to:

- Extract the information of cases and mails information from the WBCP database
- Apply raw string preprocessing tasks for some of the cases features, as explained in chapter 4.2 Dataset
- Map each case with their corresponding emails, those emails and cases with no relation are separated.
- For each case-mails relationship, create a JSON file.
- Ingest the JSON object to the database using the Elasticsearch API

4.4 Front-End: Search Engine Portal

To create the front end of the search portal, the library SearchUI was used [39]. This library is implemented in React JavaScript, provides all components to design and create a complete search experience. These layers are:

- A Search Provider, a top-level component.
- Components, that empty objects that populate the Search Provider.
- Style and Layout, which decorates the entire front-end.

4.4.1 Search Provider

The Search Provider is the top-level component of the React SearchUI. This component interacts with a deeper layer component from the SearchUI library called the Search Driver. The Search Driver component is the Headless Core of the library. A headless website architecture is a static software with no graphic interface; they are populated with information through API requests and are meant to be standalone. Inside the Search Provider, different building blocks or components can be used to personalize the user interface.

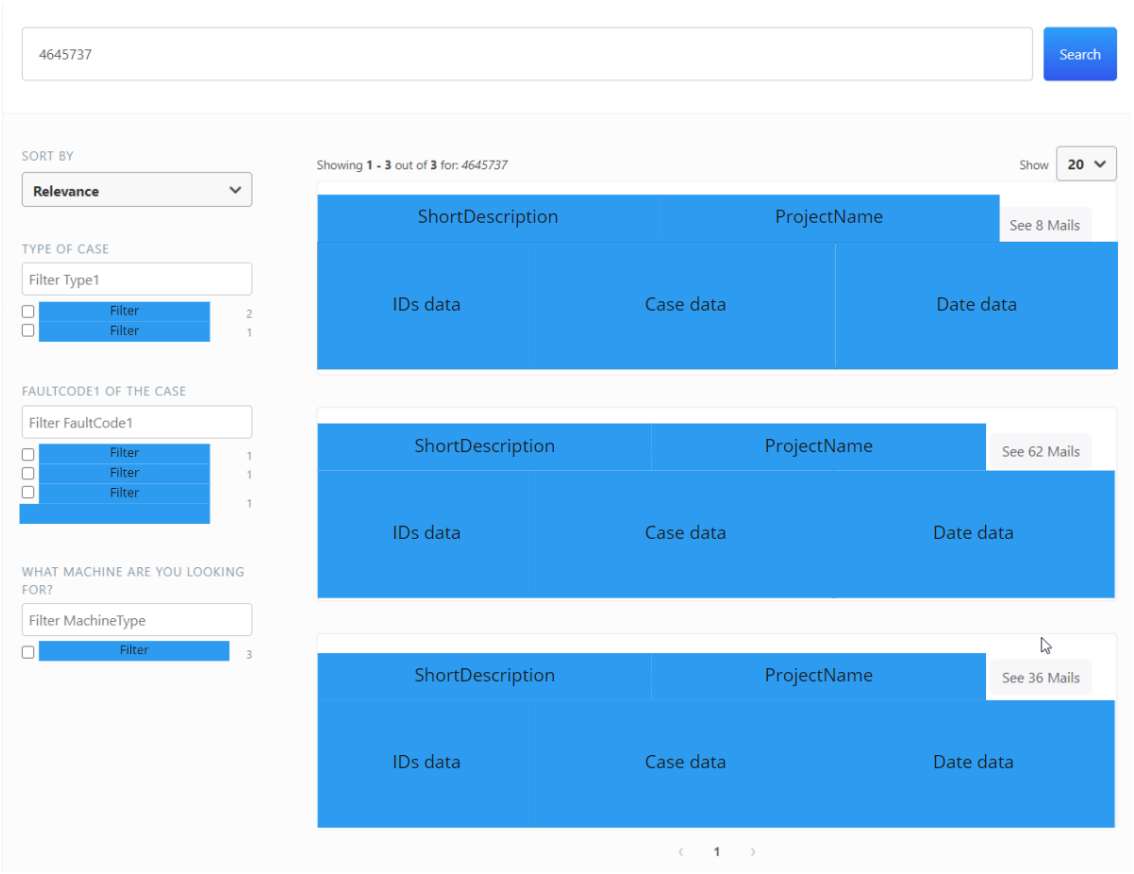

Figure 23: User interface of the search portal

4.4.2 Components

The search portal is structured in 3 sections:

Header

This section showcases where the Search Box is located, for users to introduce their queries.

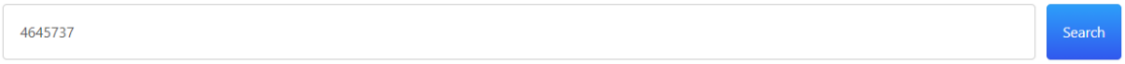

Figure 24: Search box, users can write a query to search.

Side Content

This section is where the sorting drop-down menu and the faceting/filtering lists are located

Sorting: allows the user to choose between different sorting policies of how the documents should be displayed. Three sorting policies are used in this search portal:

- Relevancy, which sorts documents by descending relevance score.
- Ascending Opening Data, sort documents by the oldest first.
- Descending Opening Data, sort document by the earliest first.

Facets: The facets enable faceting search for the user; this is displayed as checkboxes that act as filters to help to narrow down the most relevant subset of results.

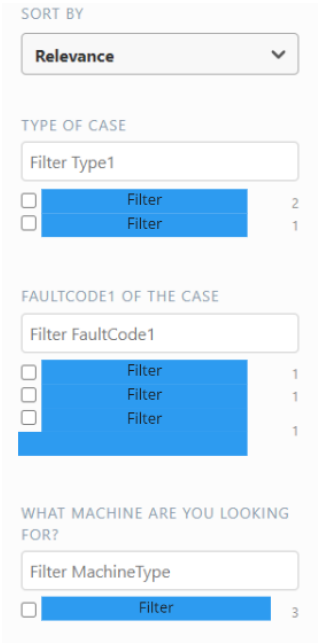

Figure 25: Sorting list and filters.

Body

This section is where the results are displayed. The section is divided again into three more blocks:

Body Header: Where the paging information is displayed, showing the number of total results for the query given. Furthermore, a dropdown menu with the options, results per page, is displayed. This dropdown menu gives the options to choose between 20, 40, 60 results per page.

Body Content:

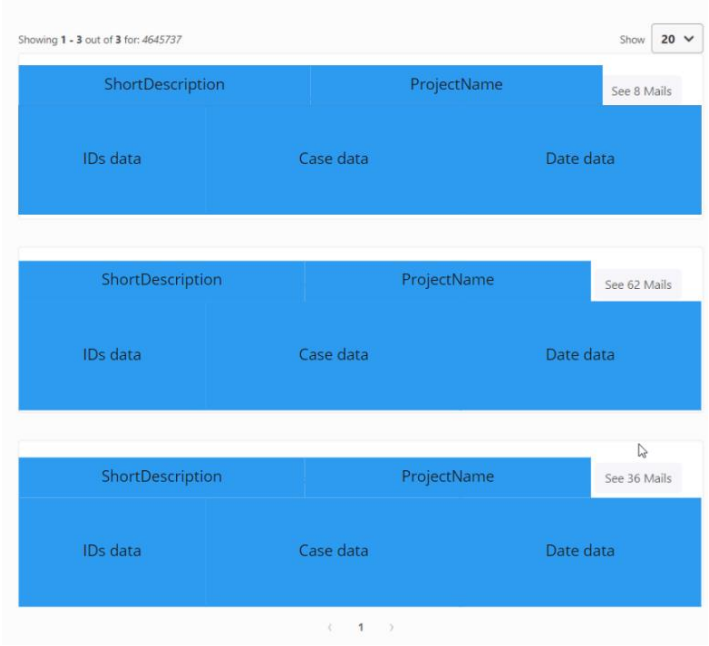

Figure 26: Results displayed in rectangles, a button gives access to the drawer

This is where the list of results is displayed. The results are represented as rectangles, which contain information about the case ordered as the figure 27 shows. The short description of the case is displayed primarily, and besides it, the name of the project the case was open. The additional information shows

The email button opens a drawer:

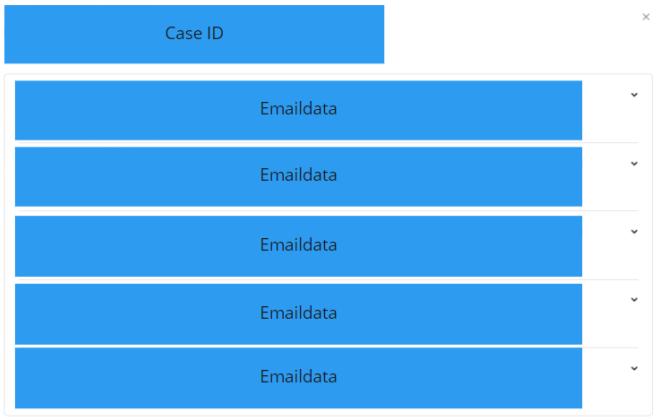

Figure 27: Drawer with the email list

which contains the case id and a list of emails represented by blocks, if the users click a block, the email body is revealed.

Body Footer:

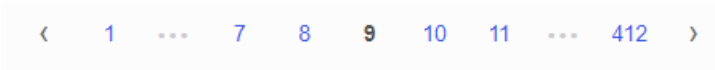

Figure 28: Pagination of the search portal

The footer is where the page navigation is located. The user can click between the four closest pages, first one and the last one.

4.4.3 Style and Layout

The style and layout that SearchUI library offers were not altered, as the objective of the project, was focused on the practical side of the portal. Another reason to not alter the style and layout was that since simplicity and clarity were the primary design objectives and this library already offer them.

Component Interaction

Now that the components are explained, to understand how these components interact with each other, a diagram is presented to support the explanation of the whole front-end environment:

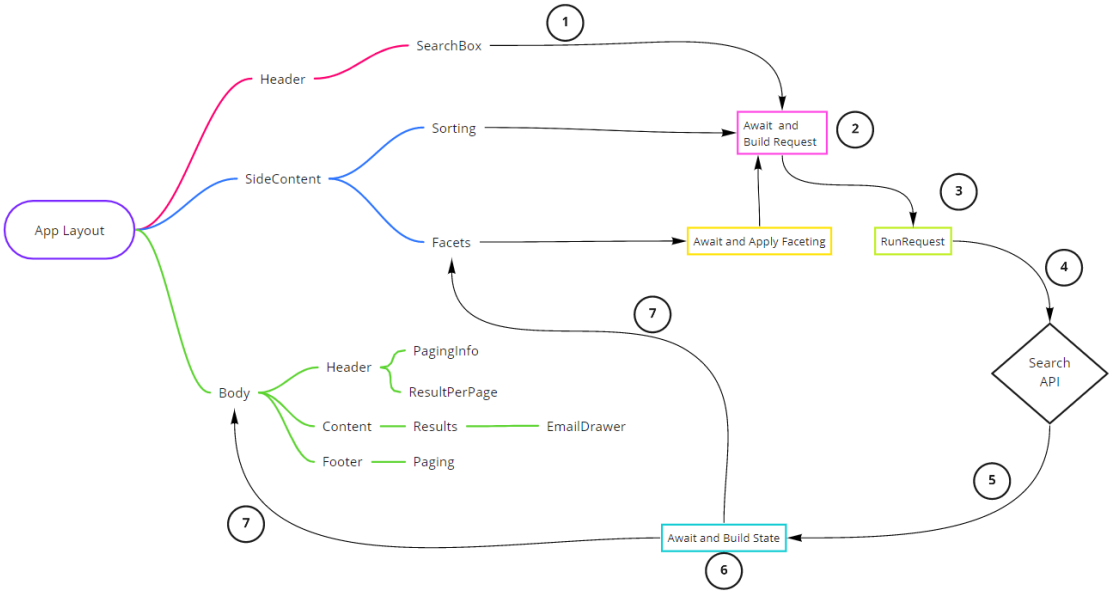

Figure 29: Mind map with steps of how data updates in different components

To understand this interaction, a step by step process of the three main functions that users will perform with the search portal is given below:

Interaction 1: User search information in the search box

1. The user inputs information into the search box.
2. The Build Request function builds an Elasticsearch JSON query object.
3. The Run Request function takes this query and formats it into a POST request JSON object.
4. The request is sent to the Search API through the Netlify Function layer.
5. The API sends back a response.
6. The Build State function captures this response and turns it into a React State to be understandable by the front-end components.
7. The components regarding the results and faceting are updated.

Interaction 2: User sets a result sorting policy

Now the user uses the Sorting component to sort the list by ascending date. This triggers another Build Request, Step 2, which takes the previous query and adds the new sorting policy. The steps 3 to 7 are executed again.

Interaction 3: User uses facets to filter the results

If the uses facets, another Build Request is trigger. Again, steps 3 to 7 are executed, and the information in the UI is updated.

As explained in the previous paragraphs regarding the interactions, the Build Request function builds a request JSON object. This request follows the Domain Specific Language or request DSL based on JSON define a request for Elasticsearch. This Request DSL JSON contains different contexts or keys-value pairs that specify different features to the request that is built and send to the Elasticsearch API. At the same time, some of the keys can be completed with a set of configurations given in the same format, producing a nested key-value JSON object for the Elasticsearch API. The request keys used for the project are:

- Highlight: to configure fragments of the data fields that will be highlighted when the query matches the fragments.
- Sort: To state the sorting fields and sorting policies.
- Size and from: To set the paginations specification.
- Source: Establish the data fields from the database that wants to be retrieved in case filtering wants to be performed.
- Aggregations: Sets which fields of the database will aggregations be performed, retrieving a list of term aggregated values for future facet filtering, this can only be performed on keyword datatypes.
- Query: To configure and personalize the queries, this field follows its Query DSL JSON format. This query format consists of:
    - A query context: to specify the query type and the fields to which the users' query will be compared.
    - A filter context: to specify the fields where the filters should be applied.

This information will be valuable to understand the next chapter 4.5 Refining the query strategy, were the development for this project of an optimal query strategy is explained

4.5 Refining the query strategy

To understand how the query strategy was tuned, it was necessary to explain first how all the components of the system are formed and communicate with each as this process considers all the elements of the system:

- The user's information needs and query habits.
- The data content of the documents.
- The Elasticsearch datatypes given to the features.
- The query functions selected and fields which the query function will search in.

The users' information needs and query habits

The users' information needs and query habits were understood through the contextual inquiry explained in chapter 2.3.1. The users' query habits showed that they query unique identifiers, machine models and problem keywords. Each of these features has their codification that the users are familiar with as they use them daily to differentiate different machines (e.g. the user knows the difference between a model AMG from an AMZ). This fact indicated that the users know exactly what they are searching for.

The users read the email body once they have found a relevant case. A relevant case is found once they have examined and confirmed that the results meet the case features they were searching.

For these reasons, the users are more interested in searching for the cases features that define each of them.

The data content

It is important to note that a significant amount of individual email body data field contains chain conversations of emails between different parties that do not need to be between the sender and receiver that the email has. This means that inside an email body, many different parties are exchanging information. As a consequence, some emails contain a chain of conversations, while others contain a unique email body.

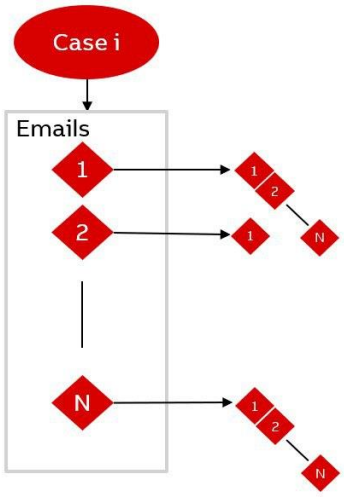

Figure 30: Representation of the email chains problem

From the point of view of a document representation, the email bodies are an unreliable source to search for information for a case because the email bodies will sometimes contain a significant amount of keywords which the BM25 algorithm will ignore and sometimes a small number of keywords.

Elasticsearch data types selected

Once the two previous points were realized, the structure of how the Elasticsearch database was created, the database shall only store as text those fields that contain free-speech data. This would make Elasticsearch store text data and keyword data in separated Lucene indexes. The Elasticsearch documentation recommends this, as keyword data retrieval is much faster as a result of the search engine using a Boolean scoring function instead of Probabilistic model function [40].

Additionally, the dates format needed to be specified as the Elasticsearch default data format was not the same as the one for the dataset. This will allow the ascendant and descendant dates sorting policies.

A query selected and fields to be queried

Considering all the last points, the most optimal query was Boolean.

A boolean query is an efficient query method when the users will search for exact matches. The main downside of the query used is that it will retrieve bad results if the users do not type the queries well. During the contextual inquiry, it was observed that the users are experts with the query terms they are used to search. Therefore, assuming users will always write queries correctly, and users are knowledgeable on what they are searching for, a Boolean query is a useful and efficient method to retrieve data.

A limitation assumed for this approach is that all the fields where the query will be compared with will have unrelated data; this means that each document field will need to have a different codification.

In Annex 2, the query used is provided to understand how the query body needs to be built. All of these elements needed to be taken into account to design the query strategy. In the following chapter, the strategy of the evaluation solution is provided

Chapter 5

5. Evaluation

5.1 Purpose

The success of any service depends on how easy, valuable, and enjoyable the entire service experience is for the users. User Experience and Usability are incredibly important aspects when designing any service. If a website is hard to use, users are likely to abandon it and search for alternatives. This also applies to search engines, as this tool has the potential to be a pillar in the users' daily work life due to the daily necessity for information in users work life, as explained in chapter 2. Due to this, it is of vital importance for any search engine to evaluate these User Experience features as close as Retrieval precision.

To make a search engine precise and relevant, special attention needs to be put on the scoring functions and query strategies that the user will follow. To make this search engine portal enjoyable to use, emphasis needs to be made on increasing the learnability of the application. The search engine portal must be easy to learn in a fair amount of time. Customer Support Units have different user groups and people of various ages. Also, since most users engage with the search engine portal to understand customers problems quickly and reply to them on the same day, it is essential that the user:

1. finds information swiftly,
2. carries the tasks without difficulty,
3. has an enjoyable experience when using the application.

In this chapter, a plan for studying the user-based search engine portal is presented. Additionally, the interactions between the users and the portal are investigated to create future recommendations. The goal is to identify the essential aspects, which need to be taken into consideration when developing the usability and user experience of the search engine portal. The approach taken is to conduct an individual inspection of the application first to understand strong and weak points. After that, users are involved in testing to eliminate potential personal biases and focus the future development in the direction users prefer.

5.2 Personal Inspection – Heuristic Evaluation

This section presents a personal inspection, conducted using heuristic evaluation. It is an efficient way to discover the most crucial problems according to existing usability principles. The 10 Heuristics for User Interface Design defined by Nielsen was used [41].

The use of Nielsen Heuristics evaluation in search engine user interfaces is not new; Ahmed et al. (2013) used these heuristics to evaluate the interface of the Web of Science, finding both strengths and weaknesses in the design of the portal [42].

5.2.1 The Benefit of Heuristic Evaluation

Evaluation is a crucial process in human-centred design. It can confirm how far the user's and organization's objectives have been met as well as provide further information for refining the design [43].

Heuristic evaluation is a technique where one or more usability and task experts review a system prototype and identify potential problems that users may face when using it. The main advantage of an expert appraisal is that it is a quick and easy way to obtain feedback and recommendations [43].

According to Nielsen's research, a handful of evaluators can find the most potential usability.

5.2.2 Nielsen Heuristic

Nielsen Heuristics describe the main aspects that need to be considered when designing a user interface and are used by usability experts to evaluate UI and spot potential issues.

In the following section, the heuristics are defined:

1. Visibility of system status

The first heuristic ensures that the system should always keep users informed about what is going on through appropriate feedback within a reasonable time.

2. Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedom

Users often choose system functions by mistake and will need a marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4. Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

6. Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. Flexibility and efficiency of use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9.Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be accessible to search, focused on the user's task, list concrete steps to be carried out, and not be too large [41].

5.3 Objectives

Based on the insights from the heuristic evaluation, three objectives of the current study were selected. Those proved in the heuristic analysis to be the most relevant (and cover the most affected areas in the portal). These objectives are, based on the insights from the heuristic evaluation, three objectives:

1) Measure Precision@K

2) Assess user satisfaction

3) Find improvement points for learnability

All are applied for the core process of the search engine portal – The old relevant case search process.

The classical metrics to measure retrieval relevance are precision and recall. However, as stated before, user queries provide thousands of relevant results, and a small number of users will be interested in all of them. This has induced the metric recall to not be relevant anymore in modern information retrieval [17]. Therefore, Precision at k documents (P@K) is a more popular metric used in the current state of the art of information retrieval engines. For example, P@10 corresponds to the number of relevant documents from the first ten documents retrieved.

Usability itself is a highly user-centered component of UX design, referring to the ease and efficiency in which a user can access or navigate a platform, product or website [44]. As digital solutions become increasingly complex, usability has become one of the most critical aspects of ensuring the success of a digital product. One of the five quality components of usability - learnability - is defined as "the time and effort required to reach a specified level of user performance with the system (also known as ease of learning)" [45]. Furthermore, people usually engage and interact more with a product or device after they understand how to use it and bounce away from products that are not well designed.

The scope of learnability will thus be first-use learnability, that was selected specifically because of the need to reflect the usage details of a new tool and create an improvement strategy (since the first impression of the search engine portal will play a significant role in whether the portal will succeed or not). The metrics used to assess first-use learnability are metrics based on task performance [46], such as:

- The percentage of users who complete the task optimally
- Number of times users asked for help
- The time until the user completes a specific task successfully.

Additionally, user experience is essential because it tries to fulfil the user's needs. It aims to provide positive experiences that keep a user loyal to the product. User experience involves the users in the development process. Considering the prior heuristic evaluation, it is assumed that this is the case and wants to validate it. It is the reason why user satisfaction was chosen as an objective. It goes beyond usability and considers the deepest motives of why users choose a digital product and stick

with it. For assessing user satisfaction, it is necessary to consider hedonic and pragmatic usage as metrics for user experience. Pragmatic aspects include personal efficiency and effectiveness, while hedonism encompasses criteria like aesthetics, joy-of-use or attractiveness [47].

5.4 User context of the evaluation study

The primary user group of this search engine portal will be the Customer Support Engineers of a manufacturing company, as the investigations were done in chapter 2.1.1 Study of the pipeline showed that the primary use of the portal would be for searching through the knowledge base of old cases that the customer support unit has already solved. Therefore, the portal will be provided as a tool to search for customer support cases relevant to the one that the users will be trying to solve during the search process.

5.5 Key issues and Questions

To validate and evaluate the evaluation objectives, the focus will be put on three main topics:

- The improvement of retrieval relevance
- The improvement of learnability
- The assessment of user satisfaction.

The classical metrics to measure retrieval relevance are precision and recall. However, as stated before, user queries provide thousands of relevant results, and a small number of users will be interested in all of them. The relevant key question regarding relevance is:

- Are the top K documents retrieved by the engine relevant for the users?

The metrics related to learnability include consistency, error recovery, task completion time, memorability. Considering the learnability aspect, we will target the following fundamental question for users:

- Is the product easy and intuitive to use, especially for first-time users?

The selection of this question is based on first-use learnability, which considers how easy it is for users to accomplish a task the first time they encounter the interface [48].

User satisfaction is another crucial issue. User satisfaction is defined as `the sum of one's feelings or attitudes toward a variety of factors affecting that situation' [49]. The relevant vital questions are:

- Does the user feel good about the search engine portal processes?
- How does the current system satisfy the user's needs?

5.6 Usability and UX Evaluation Methods

Exploration of Evaluation Methods

The evaluation methods that are used in this usability evaluation study with users are usability testing and observations. They have been specifically chosen to measure the precision of the retrieval engine for the users, to assess learnability and user satisfaction as well as reveal the metrics from the introduced in the Objectives. All methods mentioned are used to cover UX and usability evaluation through triangulation. Triangulation is a method that aims to strengthen the validity of the results by utilizing different methods to collect data on the same topic [50].

5.7 Remote Moderated Usability Testing

Usability testing with real users is “the most fundamental usability method and is in some sense irreplaceable since it provides direct information about how people use computers and what their

exact problems are with the concrete interface being tested." [51]. The method involves users interacting with the platform. The outcome of the method is to point out participants' struggles when they use the portal, with facilitators observing their reactions closely. The method has proven efficient, for example, in assessing Washington State University System's issues [52]. The deliverable of the study mentioned above was a table with a task, a problem, an owner, and action to be taken to solve the problem - that was successfully used to improve the application.

An option of usability testing is remote moderated usability testing. This method is proposed due to low resources required and high flexibility in terms of timing. Remote moderated usability tests work very similarly to in-person studies. A comparison by Andreasen et al. has shown that remote testing is "[. . .] virtually equivalent to the conventional method." [53], as they lead to the same results. The facilitator still interacts with the participants and asks them to perform tasks. Usually, moderated tests can be performed using screen-sharing software like Zoom, Skype or GoToMeeting [54]. Tasks are executed throughout the remote usability testing. A task scenario is an action that is asked to the participant to take on the tested interface. The task needs to be realistic, as something that would happen in a real scenario in the portal. It should be phrased in an actionable way, so the participant does not ask questions, but rather performs the action. Lastly, it should require as little intervention from the facilitator as possible in order to avoid biased results [55].

To better understand the drives of the participants while they are executing the tasks, we use the Thinking Out Loud method. The method implies that participants are asked to think out loud as they are performing a task. Users are asked to say whatever they are looking at, thinking, doing, and feeling at each moment. In this way, the facilitator still has access to the participant's logic of thoughts and can follow their reactions and actions, even though not being physically close.

5.8 Observations

Observational research is a simple, yet demonstrated effective method, in which evaluators observe users interacting with the portal. This helps understanding, from an objective point of view (not a personal user opinion), how users use the portal, as well as their real-time reactions. The method has been used in the past to reveal how consumers are using the web and what patterns they reveal through their actions [56]. The results from the previous study have been curious and against the original focus group comments, stating that they usually search the web to find detailed and trusted information. In reality, “None of the participants actively searched for information on who stood behind the sites or how the information had been compiled; often, they did not even visit the home page." We hope to also validate through observations if the previous methods' results can be trusted.

Observations can be used in remote moderated usability testing to observe participants' reactions, mimics, and actions, and validate if those match the results of the other two methods.

5.9 Testing Proposal

Remote Moderated Usability Evaluation (RMUE) is proposed combined with Observations made by the evaluator. Participants will conduct at least three individual RMUEs. There should be at least a couple of weeks between the participant’s RMUEs to ensure realism. The participants are given the same tasks in each RMUE. The tasks are part of the” Searching for a relevant old case” process, as that is the focus of the evaluation.

Participants are asked to think out loud during task completion, and their performance is observed and measured so that it can be compared over time. Overall, —across the participants—the metrics of learnability, mentioned in Objectives Section of this chapter, are measured. After three sessions

with each of the participants, it is assumed that enough data is accumulated. However, if the conductor of the evaluation believe that the findings are not saturated, the session can and should continue.

As an additional perspective, each user will be asked to test a query in the previous tool used, so that the time completion achieved in the new prototype can be compared to a relevant score. A time limit of 120 s is set. This is the maximum time a task should take.

5.10 Participants

The focus of the study plan is on the customer support engineers unit where the project has been done. The ideal participants for the study are fluent in English and are comfortable with using technology. The participants must be able to use video call software as part of the evaluation study.

The participants in the study are required to be first time users of the Search Engine portal and not to have any previous experience from using the portal. The first-time participants are essential for the study to be able to assess the learnability of the study participants. The focus group for the study should include both female and male participants. The portal aims to serve users of all ages. Participants should take part in the study based on their interests and availability. For usability testing and interviews, the participants need to seek for is at least 5. The 5 participants will take part in 3 usability evaluation sessions. For the interviews, it needs to be decided if 5 participants are enough to reach a saturation point of the interview answers or if more participants need to be recruited later. For the survey and to carry out statistical analysis, such as the t-tests, the number of participants should be as high as possible, preferably at least 100 respondents, to have statistical significance. Nevertheless, since the customer support does not have that many people in the unit, it will be sent to all of them to reach the majority of the department.

5.11 Planning & Conducting Remote Moderated Usability Testing

In this section, the testing procedure and additional details are explained:

5.11.1 Session Planning

Because the evaluation is remote, the location of it is relatively flexible. It was advised that both parties—facilitator and participant—are in an environment where they are not disturbed through other people or noise.

While the platform for the remote call can be seen as a matter of taste or convenience, basic features should include recording meetings and screen sharing. The facilitator should be comfortable with the software and able to help the participants if needed. The facilitator and the participant should ideally both have headphones and a proper microphone to ensure clear communication.

As the number of participants is relatively small, a simple option for recruiting is asking Customer Support Engineers directly via email. Noteworthy is that the participants can read English, as the evaluator can only English and Spanish.

Pilot Test

Before establishing a session plan for the RMUT, a pilot test was conducted to ensure the validity and reliability of the evaluation plan. The session was conducted with one user with the objective of testing whether the instructions planned where prone to any potential confusion for the users.

The recruitment of the user was performed by randomly selecting a potential user from the customer support unit, contacting the user via email. The user has been working for the unit for more than five years, responsible for solving customer support issues in the unit.

The session for the pilot test had an open-ended approach, with no fixed tasks given by the facilitator to the user, but rather looking at how the user formulates tasks and what he naturally searches for to observe patterns and logic.

The facilitator would then explain the route map of the test: introduction, explanation of the two tasks, tasks execution and a test debrief.

In the test introduction, the facilitator made sure the user is comfortable by establishing rapport. The facilitator confirmed if the user meets the requirements established for the test about being alone in a room and ensuring a 30-minute gap of no interruption.

In the explanation, the facilitator asked the user to remember two recent Customer Support case and think out loud how would he search for it. After the user explained to the facilitator how the query would be done, the facilitator presented the interface of the search engine portal. Additionally, the facilitator would give control to the user.

In the tasks' execution, the user started using the search portal to perform the tasks, explaining out loud the train of thoughts during the whole process. After each task finalizes, the facilitator asked for feedback about how it was the process.

The questions asked were:

- Did you find it challenging to perform the tasks?
- What challenges did you find?
- Rate each task in terms of difficulty on a scale of 1-5 (1 – the easiest, 5 – the most difficult)

- Did you like the app?

Afterwards, the facilitator will ask the user to perform the same tasks in the old knowledge tool that the users use to find the data used in this project.

Pilot test results

The pilot test proved to be an excellent first step to learn and understand the strengths and weaknesses of the planned RMUT.

The test introduction went smooth, and the facilitator appeared to establish a successful rapport, as the user sounded comfortable and relaxed during the explanation, the instructions where clear as the user successfully expresses to the facilitator how he would execute the query in the search portal.

During the execution of the tasks, the user needed some help to understand better how the application worked. As the RMUT is a moderated test, the facilitator helps the user with the small questions that asked.

The answers to the question turned to be short and non-descriptive. This problem was due to the lack of preparation, as a mistake of the facilitator. The questions needed to be open, to give space to the user to describe better the design, as stated in [57]. Therefore, new questions were generated and are presented in the next section, Task Design.

The debrief was also smooth; the user expressed the satisfaction of accessing information quickly and compared it with the old tool used in the customer support unit. The facilitator detected a deficit of critical feedback in the pilot test. This is common now in the UX researching field as Friendliness bias: "*Friendliness bias — also called acquiescence bias or user research bias — occurs when the people providing feedback tell you the answers they think you want to hear. Sometimes, this happens because they think fondly of you and respect your professional opinion, but the reason can also be less flattering*" [58]. To avoid this bias, in the introduction, the rapport established with the user needs to be measured not to appear too friendly.

Thanks to this exercise, a whole design iteration was performed that served to enhance the main RMUT.

5.11.2 Task Design

To perform the test, the three following tasks are proposed:

(1) Search for a warranty case of a specific machine with a specific problem
(2) Search for the most common problem for a specific machine
(3) Search for the cases using a unique machine identifier.

The following scenarios are examples of what can be given as goals to participants.

Scenario 1 "You are working on a case regarding a problem code from a specific machine model. You remember that four years ago, something similar happened to another customer with the same machine. Search for all cases with specific machine model and problem."
Scenario 2 "You are working on a case where the customer has a problem with a machine model, but the client does not give enough information. You want to check what are the most common problems for that specific machine."
Scenario 3 "An old client has a problem with a Generator; the client only gives a machine unique identifier. You want to check for the problems in the past of these machines to understand the current problem better."

At the end of each task, take some time to ask questions. If nothing notable happened that could be further explored, ask the following questions:

- How did the process go? Please describe in detail.
- What challenges did you find?
- Rate each task in terms of difficulty on a scale of 1-3 (1 – the easiest, 3 – the most difficult)
- What positive experiences did you have?
- From the ten first results, which ones are relevant for you and which are not?

Session Design
The session comprises an introduction, the tasks, and a debrief.

In the beginning, during the introduction, the facilitator makes sure that the participant is comfortable and that they understand what will be happening in the session. The general outline of the session, and what is expected of them is introduced: The session takes about 30 minutes. The participants go through three tasks, during which they are asked to think out loud. After each task, there are a few brief follow-up questions.

If the participant has no further questions the participant, and the consent form is filled out, recording starts and the transition into the scenarios can begin. The participant is asked to enter the search engine portal, and the facilitator presents the first task. After all the three scenarios, the facilitator sums up the evaluation by giving space/asking for feedback and thanking the participant for their help.

The facilitator should follow some general guidelines not to influence the results: Use clear instructions, watch for verbal cues of participants, do not speak too much, let the participant be in charge of the task, and avoid voicing opinions. It is advised that the facilitators prepare for the evaluation by reading about usability testing – A good starting point is the "Usability Testing 101" blog article by the Nielsen Norman Group [59].

5.11.3 Measures & Evaluation Criteria

In this section, it is defined as the evaluation criteria for measuring outcomes based on our study objectives and introduce the methods for data elicitation and data analysis.

5.11.4 Outcome measures and evaluation criteria

Defining specific measures and evaluation criteria is required for the evaluation. It is helpful and essential to measure the outcomes of the study and target study objectives. Together with how to measure learnability and user satisfaction, threshold values for the evaluation criteria are defined as follows:

Measurement of Precision@10

The measurement of Precision@10 considers the relevant number of documents for a given query. It is a quantitative measurement that has to be done for the same tasks trying different queries strategies. Precision@k has the advantage of not requiring any estimate of the size of the set of relevant documents but the disadvantages that it is the least stable of the commonly used evaluation measures and that it does not average well, since the total number of relevant documents for a query has a strong influence on precision at k [16].

Measurement of Learnability

There are two specific outcomes defined to measure learnability. The first one is how long it takes for users to become proficient in accomplishing a particular goal or task after using the Search engine portal for the first time. The related evaluation criteria are the amount of effort and the times of task sessions [60]. The threshold value is defined as the learning curve to describe Effort (y-axis) / Goals (x-axis). Because the standard way to analyze and present learnability data is to focus on one performance metric for a specific task, time on task is calculated as a metric to quantify effort. Completing a task like booking an appointment is the goal. The same participants needed to complete the same task multiple times. It is recommended repeating the task until a plateau is reached. A flattened curve indicates that participants have learned the system (specific to this task) as much as possible [61].

The other one is how easy for new users to learn to operate the interface. The related evaluation criteria are errors, number of task steps, and task success/failure. The threshold value is defined as: how many steps does the user need to operate for each task? Does the user complete the task successfully and accurately? How many errors does the user encounter during the performance of the task?

There are two specific outcomes defined to measure user satisfaction. They are related to critical questions for user satisfaction. The first one is to discuss: does the current system satisfy the user's needs? The evaluation criteria are understanding the gap between customer expectations and performance perceptions [62]. Does the threshold value include customers' potential needs and pain points?

The second one is about: does the user feel good about the Search Engine Portal? For this one, customer satisfaction scores with the quality, efficiency of the product are the evaluation criteria.

5.11.7 Methods for data elicitation and data analysis

There are different methods for data elicitation and data analysis based on the primary method we plan to use for testing objectives. For improving learnability, usability testing was conducted. For assessing user satisfaction, interviews during the usability test and observations were used.

Methods for Learnability

Collecting learnability data is the same as collecting data for other metrics. However, the analysis would only include metrics that specifically focus on efficiency. It is suggested using" trials within the same session" method for data elicitation. The session indicates the task planned to test, which is looking for old information, using different queries. It is a method that the user performs tasks, or task sets, consecutively, without interruption. This method does not consider memory loss [60].

Quantitative data is analysed through comparison of the task means in terms of success, difficulty and time.

Methods for User satisfaction

Interview during usability testing and observations are two commonly used methods for data elicitation. There are both quantitative and qualitative data that can be collected from the RMUT and Users (or potential users) can, with the aid of pairs of opposite adjectives, indicate how they experience your product [63]. Otherwise, there is more qualitative data coming from observation methods.

For quantitative analysis, data means comparisons can still be used as a data analysis method. For qualitative analysis, affinity diagrams can be used which is an excellent method to help make sense of all information for mixed data, such as facts, ideas from brainstorms, user opinions, user needs and insights [64].

5.11.8 Data collection planning

In this part, the data collection is planned. There are some aspects included, such as methods, settings, types, and the number of cases.

Plan for Precision@K

The data collection method for measuring Precision@10 is asking the users whether the first ten documents retrieved are relevant or not for the scenario presented. This data type is quantitative. Each time a user will be evaluated, a different query strategy will be used; this will only influence the documents retrieved and will not have any influence on the user experience.

Plan for Learnability

The data collection method for measuring learnability is trials within the same session. In the usability testing plan, we need to design the task scenarios, go through them, figure out the number of steps for completing each. The data type is quantitative data. The number of cases would be several trials, depending on how users finish the goal.

Plan for User satisfaction

The data collection methods for measuring user satisfaction are interviews during RMUT and observation. In settings, for interviews, the facilitators should ask a set of predefined questions after each iteration. For observations in a small way, the facilitators should check the software environment first. In this usability testing plan, the data type is quantitative data and qualitative.

5.11.9 Analysing Results

The data collected from our evaluation consists of two parts: the quantitative data from the usability testing, and the qualitative data from observation and interview.

Quantitative data offer numerical metrics of the usability of design and precision@10. Regarding the usability results they can be based on users' performance on a given task (e.g., task-completion times, success rates, number of errors) or can reflect participants' perception of usability (e.g., satisfaction ratings). When evaluating the quantitative data, statistical significance is critical validation. Mathematical instruments such as confidence intervals and statistical significance can tell how likely it reflects the truth or the effect of random noise. Quantitative data can tell that the design may not be used relative to a reference point, but they do not point out what problems users encountered [65].

Qualitative data represents people's thoughts, beliefs, self-reported needs and behaviours about the object. Qualitative data can be used to understand the specific usability issues in an interface.

5.11.10 Ethical issues

Our evaluation highly respects the user's emotions and well-being. The testing will be conducted under the ethical guidance of Nielsen. The critical ethical issues are including but not limited to [66]:

- The purpose of the research. It is the system that is being tested, not the user.
- Who are the members of the research team
- Participants can stop any time during the study
- How the data will be used in research. The identities of users will not be revealed to any outsiders, according to General Data Protection Regulation (GDPR).
- Explain any recording, or other monitoring that is used

These ethical issues are explained in an informed consent form. Before the test, every participant will receive a copy of informed consent, read through it, and sign if they agree.

Chapter 6

6. Results

In this chapter, the evaluation of the system is presented and elaborated through a series of testing scenarios and methods, previously presented in chapter 5. While the latter chapter presented a more theoretical framing of the evaluation methods, the following chapter is from a practical perspective, applying the theory through individual and collective testing to get a comprehensive and realistic dimension of the system's capabilities.

6.1 Personal Inspection – Heuristics Evaluation

The heuristics evaluation served as a good starting point to have a critical point of view of the project. In this section, for each of the ten evaluation points, comments are given if the design rule is applied or whether it is violated.

Visibility of system status

For the tested system, the evaluator considers the rule is being applied to the argument that the results count are updated and displayed in real-time after the user applies a search filter. In the same process of querying, the real-time update of the filters is easing the user's effort to see the filters applied. The interface provides the interaction needed for the "See Emails" button, which changes colour when the mouse cursor hovers on top of it.

What poses challenges for the visibility of system status is the current inability to change the filters needed, since all filters previously checked are activated upon selection. If a considerable amount of filters are activated, the user can lose visibility on the current filters applied. The last point observed through the heuristic evaluation is that the main page of the system does not offer information on what is the location of the user in the search process.

Match between system and the real world

The system's language is framed for a customer support role, with relevant terms, used in a real-life scenario (e.g. failure code, machine, case number, serial number). This presents advantages while being used since it offers the familiarity desired to users. The search output also helps in the process, by showcasing the number of results in a query, thus providing an overview of the whole system instead of only showing isolated cases.

On the other hand, if the user's goal is to search for sub-set cases, for a specific type of problem, the search is conducted with individual keywords, generic keywords being impractical.

User control and freedom

The main search box, which is the core functionality of the prototype, is strategically visible and available on the upper part of the interface, offering the all-time control needed to formulate queries.

A drawback for users can be the filters list which does not have the functionality of being deselected once activated without returning to the original position on the list.

Consistency and standards

Search Filtering Screen has a consistent design that remains stable throughout usage: there are no changes in language, design, or functionality in the querying process.

While overall the design is consistent in the *Search Filtering Screen*, the prototype first loads with a different, more minimalistic page design, that does not have any filtering functionalities. This different design can confuse users and pose challenges to the overall unity of features.

Another confusion can appear when a case does not have a case description: the space occupied by this description disappears, and the next component takes its place. This inconsistency can provoke confusion as the user might recognize the project name as the short description of the case.

Error prevention

Users can apply and discard filters as they need to get closer to good search results.

This heuristic, however, highlights a possible weak link in the search chain when the user mistypes a query: the system does not yet offer a possible correction of the error. This step can cause frustration since not being able to formulate a query correctly leads to incorrect results.

Recognition rather than recall

This heuristic highlighted that the filters deselection from *Visibility of system status* findings has an impact on recognition as well: users might not be able to easily recognize which filters are activated and which not (problem emphasized by the usage of numbers in the filters' name, that are hard to compare and differentiate).

Flexibility and efficiency of use

The existence of filters and the possibility to sort offers users the flexibility desired and easiness of finding a query result. While there is space for improving the search algorithm itself, overall, the system is highly flexible and adjusted to support enough personalization of user's querying needs.

Aesthetic and minimalist design

A strong point of the prototype is its minimalist design: there is no cumbersome information present in the interface. The user is presented with the core functionality of the system from the beginning – the search bar – accompanied by two other functionalities that enhance it: filtering and sorting. Case details are compressed in the" See Mails" button and available only at need, which highly simplifies the interface and showcases only what is needed.

Help users recognize, diagnose, and recover from errors

Recognizing and recovering from errors is a challenging aspect of the interface. Users are not shown error messages nor receive guidance on how to fix a mistake once committed.

Help and documentation

The filters search box provides an opportunity for documentation since it offers helpful labels with an explanation for each filter.

What can be considered a drawback is that the platform does not seem to be addressing unexperienced/first time users' needs. This again enforces the importance of learnability when designing a new tool; finding what works and what not for first-time users can offer possibilities for speeding the learning process and consequently, making the tool usable.

Conclusion

The heuristic evaluation concluded that the prototype's strong points are its minimalist design, efficiency of use and matching of the real world's needs, while improvement needs to be done for handling errors (from recognizing to recover from them).

These are currently assumptions and need to be tested with real users. The heuristics evaluation offered an internalization opportunity: only by diving deeper into the system through the empathizing lenses of a user one can gather more insights and an objective perspective of the system's strengths and flaws. This exercise however is not a replacement of real users' feedback, rather an intention for an impartial and fair individual evaluation. The heuristics prepared the ground for testing the assumptions discovered and uncovering the root cause of agreement or disagreement. Only when the system's builder understands the similarities and differences between his and the users' thinking he can further build a solution that fulfils real needs.

The next chapter explores the users' thinking process through remote moderated usability testing.

6.2 Remote Moderated Usability Testing

To analyse the RMUT results of the methodology described in chapter 5.6, a comparison is made between the means for the three variables followed: percentage of success for a task, percentage of success without help and task completion time. The users' comments were also captured and analysed to reflect their opinion on the prototype (exact comments are referenced in Annex 4). The facilitator observed the participants' behaviour throughout the RMUT to explore potential contradictions between what they think-aloud and what their actions reflect.

Percentage of users who succeeded in finding their information need

Before concluding, one needs to understand the definition of success in the context of information retrieval for the customer support team involved: succeeding to retrieve the information they need requires finding one relevant information in the search engine. For the task-completion rate (with or without help), an average of 78% was taken as a threshold for measuring success [67].

All the users succeeded accomplishing the three tasks and finding at least one relevant information for each of those, emphasizing a positive result of the tool.

Percentage of users who succeeded without any help

Throughout the tasks, users reflected different behaviours for searching in this variable.

Task 1 and 3 revealed a 100% success of retrieving the necessary information need without any help. The assumption made in 4.5 was observed and considered valid in this task (although this does not mean the assumption is universally true): users seem knowledgeable. They know how to search for technical information for their cases. Tasks 1 and 3 were also assumed to be easier than task 2, since they required a one-step search process, while task 2 required more complicated actions, not explained to users.

It was observed therefore that task 2 posed difficulties to the majority of participants and obtained a result of 40% success (meaning 60 % of the users required help in searching for complicated queries). Users spend more time exploring the interface with the cursor, searching for potential guidance from the system; when help was not found, they relied on facilitator's help to offer solutions to task completion.

The overall score of the average percentage of users who succeeded without any help was 80%, which is considered successful compared to the previous task completion rate of 78%.

Completion time

The same pattern is observed when measuring this variable: tasks 1 and 3 have smaller average completion times (20.4s and 27.6s respectively), observations telling this is being caused by interface exploration. Task 2 obtained a completion time of 35.8 s since users spent more time asking questions and deducing the role of filter and sorting interface components.

The overall score of the average completion time was 27.93 s. The time obtained is considered a good one if compared to the previous tool's query completion time, which was tested and achieved a result of 60.53s. The previous tool used by the customer support unit was not able to provide a fast resolution for task 2, which provoked that users exceeded the time limit of 120 s.

Qualitative comments

The RMUT sessions were additionally expanded by participant's comments about the session and tasks.

From these comments, the general impression of the system is a positive one: "The process went well", "The process was easy", "Process was not difficult".

Participants did mention critical opportunities for improving the prototype, referring to adding more documentation for first-time users ("the main challenge was to figure out what the filter represented without previous explanation"), as well as improving the filter functionality ("it should be more useful to have a specific filter only for the machine model and another for the machine specification code") and making the search process to support users' flexibility and efficiency of use ("The filtering is good but it should have a general button to undo all changes").

6.3 Effectivity and performance

The effectivity of the retrieval was measured using precision@10.

This measure is considered a relevant one in this context since by observing users in their contextual inquiry it was assumed the position of the relevant documents does not matter, as long as a relevant document are retrieved in the first set of 10 documents. As an example, it was not considered necessary for the first result to be relevant if at least one of the next nine documents was relevant and displayed on the interface.

Each participant's task precision results were measured, and then the average of the three tasks was computed. The first task precision result showed an average of 82%, while task 2 resulted in a precision of 94% and task 3, a precision of 100%. Overall, the system scored a precision@10 of 92%, which was rounded to 9/10 relevant results in the system (since the results are considered Boolean). The precision is considered a positive outcome which reflects an effective retrieval of relevant results of the prototype.

Table 11: Quantitative results of the evaluation

	QUANTITATIVE														
	Relevance			**First use learnability**											
	Precision@10			**% success for task(Bool)**			**%success w/o help(Bool)**			**completion time (s)**			**old completion time(s)**		
Users	Task 1	Task 2	Task 3	Task 1	Task 2	Task 3	Task 1	Task 2	Task 3	Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
User 1	0,8	0,9	1	1	1	1	1	0	1	24	42	30	37	120	29
User 2	0,8	1	1	1	1	1	1	0	1	21	40	33	38	120	27
User 3	0,9	0,9	1	1	1	1	1	1	1	18	30	21	33	120	23
User 4	0,9	0,9	1	1	1	1	1	1	1	18	28	21	34	120	25
User 5	0,7	1	1	1	1	1	1	0	1	21	39	33	38	120	24
Avg/ Task	0,82	0,94	1	1	1	1	1	0,4	1	20,4	35,8	27,6	36	120	25,6
Total Avg		0,92		1			0,8				27,9			60,5	

Chapter 7

7. Discussion

7.1 Limitations and assumptions

Limitations

When handling current prototype limitations, it is wise to mention three perspectives hinder the results obtained: technical, evaluative, and general. From a technical viewpoint, we have explored in chapter 4.5 the existence of cases that are childless emails or orphan cases, which has caused the inability of one-third of the database, as the users cannot use this data due to the inexistent connection of parent and child. Another limitation is present in the email body data fields, that can hold more than one conversation; if in the future it is required to do a full-text search, this would not be possible due to the email body fields of the cases containing chain conversation from different parties.

The evaluation also presents some limitations. The usage of within-subjects design in usability testing exposes the research to potential internal validity problems related to history. Because participants are exposed to sequential tasks, they might learn how the MVP works, and thus the results can be biased when measuring completion time. The number of participants chosen for the evaluation (5) – mostly because it was done in the short timeframe of two weeks - is not representative if the tool seeks to be implemented on a larger scale; the short period is not enough to showcase the essence of the real users' feedback, so more time is needed to achieve the data saturation point.

Also, the facilitator's experience in applying the evaluation methods is limited, which can be a problem, especially for usability testing, where the facilitation is crucial to obtaining satisfactory results.

In general terms, the scope of thesis discussions itself can be a limitation: because the author does not have an extensive experience in information retrieval as well as in the role of customer support, this represents a challenge for understanding the real problems and information needs.

Assumptions

Throughout the research, some assumptions were made to simplify overly complicated needs, such as:

- Assuming users are knowledgeable: The interface provides technical terms, specific to customer support needs, that might not consider recently employed personnel, who does not have to experience necessary to comprehend and query similarly to the participants who have been recruited for the evaluation

- Assuming users do not make mistakes: The author considered customer support engineers are meticulous when querying. Thus the interface does not currently help to provide recommendations for errors or help in solving them. However, it is humanly normal to commit errors and not get the desired results due to this.

- Assuming users skim through the entire page of the query results (and not search only for the first k queries): From the observations conducted in the contextual inquiry and confirmed, it was assumed it is not a must-have functionality to have high precision in the first results (like the mean average precision entails a high weight for those mentioned above), but that an output of the relevant documents in the page (like precision@k does, not considering the position of the relevant documents) would suffice.

7.2 Future work

This project has proved to be a rewarding multidisciplinary work: Customer support, Data structures analysis, Information retrieval, Systems communications and architecture, User interfaces and User experience. All of them have provided challenges: some of them have been tackled as a requirement to create the project, some have opened the doors for future steps, and others have been mixed to create a more robust product.

Data

Research of email parts classification can open the door to exploit the data inside and emails not used in the dataset. Indeed, Repke and Krestel proposed in [68] a recurrent neuronal network to classify parts of email in different zones. This approach can be explored to extract information of email bodies to connect them with the childless cases, augmenting the data available.

Potential deployment

If the stakeholders would be interested in deploying the solution, the database used can be augmented with other knowledge bases with similar data. The Elasticsearch back-end scalability should be considered. To scale Elasticsearch, three different features shall be taken into account.

- The index size: how is the Elasticsearch index going to scale.
- The cluster size: how many nodes is the cluster going to have.
- The throughput: how many concurrent searches will need to be managed.

These three points are dependent on each other, meaning that increasing one can only be achieved by the expenses of the others [69].

If more data need to be ingested in the system, the index size will scale. This implies more documents will need to be stored in the nodes. Hence more documents will be stored in the shards. As stated by Ragone in [69], "the number of documents per shard gives a sense of how long will a search on a shard take. Furthermore, the number of documents per node gives an idea of the memory requirements of each node."A proposal of a future cloud deployment is given, in case the stakeholders find them relevant:

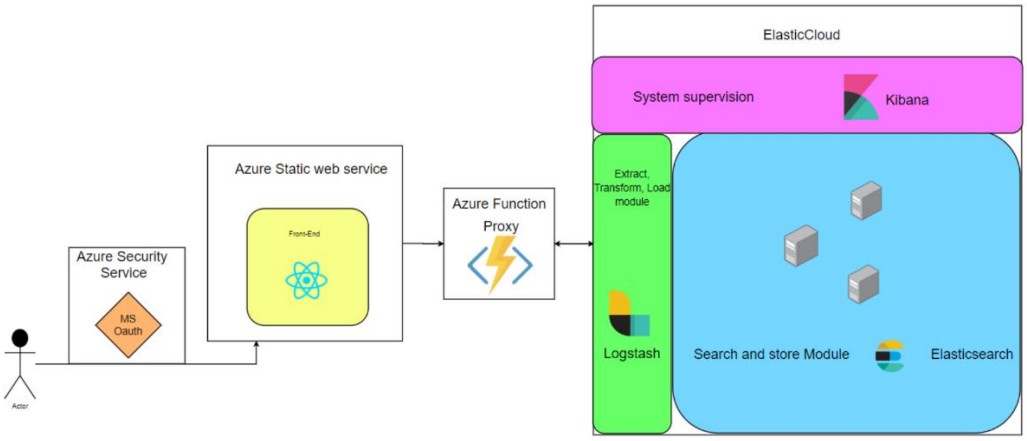

Figure 31: Future architecture for a potential deployment

This Elasticsearch deployment proposed has three nodes. The one shall be configurated as a master node and the other two as data nodes. The index shall be sharded in 3 and separated as the picture suggests. Three virtual machines shall be needed for cluster deployment.

Additionally, a Serverless function deployment is recommended, as it was explained and use, during the development. As the project was developed with Netlify, deploying the front-end on this serverless cloud provider should not be complicated.

User interface

During the heuristic's evaluation, small design flaws in the front-end were identified. Solving them can improve the overall search experience of the portal. Furthermore, another idea generated during the evaluation process was to generate a cloud of words for each email displayed in the drawer of the front-end. This cloud of words will contain keywords of the email that can help the users understand the information of the email without needing to go through it. The keyword extraction algorithm can be done before the data is ingested in the database as a data field for each email. Still, without assessing email parts classification mentioned before, the keyword extraction cannot provide good result.

Customer Support

The possibility to study the customer support pipeline for the stakeholders gave the author some ideas on how the overall pipeline can be improved.

The author noticed the low automatization in the customer support unit. Automating sections of the pipeline can be an excellent improvement to the unit. Semantic analysis techniques can extract the semantic meaning out of email corpora could open the door to:

CS engineer profiling can help to understand which worker has more expertise in specific problems. Additionally, it can lead to automated case assignment, to route cases to those workers more knowledgeable in a specific problem. Still, it can be bad for the unit, to concentrate knowledge in different workers, reducing the unit's resilience to change in the hypothetical case a knowledgeable worker cannot work, or stop working in the unit.

Knowledge graphs generated through ontologies and natural language processing technique. Ontologies are formal definitions of types, properties, and relations between entities for a specific domain. With this technology, it is possible to generate knowledge graphs of the different problems of the customer. By abstracting the problem and resolution of customer problems, this technology could potentially open the door for the automated response system for those cases which knowledge graphs achieved good accuracy. In [70], Beseiso et al. propose an ontology architecture for email knowledge extraction.

This project has opened different opportunities for achieving better efficiency in the unit, better work experience for customer support.

Chapter 8

8. Conclusions

The research aimed to explore timely opportunities to find previous relevant cases from a knowledge base quickly enough to speed up the resolution of the new case at hand.

This primary research question raised several sub-questions that needed to be answered in order to convey empirical recommendations (What are the minimum/key features that an MVP, dedicated to fast information retrieval for customer support, has; What is an optimal query strategy that can satisfy the user information needs; What are potential challenges encountered in building a search portal MVP and how can those be tackled).

Based on the usability evaluation methods (contextual inquiry, remote moderated usability testing and observations), it was concluded that a research engine portal could be a potential solution of speeding up cases resolutions. This is due to the possibility of personalizing the portal features and users' needs and providing an internal specialized tool which concentrates on suiting company's specific pipeline of information retrieval (an aspect that it is hard to be attained by using the current generic, external tool).

To build this specialized tool, the minimum vital features the empirical research highlighted as necessary are:

- Be able to provide full-stack information retrieval solutions.
- Be able to manage nested data structures.
- Be able to provide fast and relevant searches.
- Be able to integrate easily with cloud services from stakeholders.
- Be able to scale horizontally quickly.
- Be relevant enough to find online support to solve problems.

Elasticsearch technology fulfils all these essential features.The other basic requirements the tool needs to have, this time from a user's perspective, are:

- An easy to navigate user-interface
- Fast output of the system
- Relevant output of the system

The successful choice of the critical features is reflected by the measurements obtained in first-use learnability, precision@k and user satisfaction. The quantitative evaluation showed positive scores of 100% task completion rate and 80% success without any help (which are above the 78% threshold taken) and an average task completion time of 27.93s that was lower than the time corresponding to the previous tool used – 60.53s. From users' perspectives, the qualitative results embrace the quantitative spectrum, overall comments being positive while still critically assessing the needs that could be accomplished in the future (such as better retrieval – better task completion time and UI improvements – better time-completion rate without any help).

By providing a simple, clean interface, the portal managed to visually display a minimalist design, that revolves around the core functionality of the MVP: the search. Both fast and relevant output is conveyed through the additional functionalities provided - filter and sorting - which allows users to control the desired queries to be achieved fast systematically and with relevant results, as well as with a manageable effort from their side.

The third sub-question (What is an optimal query strategy that can satisfy the user information needs) was explored by observing user's habits on one side, and the data intricacies on the other side. The output that resulted from these observations was a 4-step strategy, meant to discard cumbersome data (through Elasticsearch) and reflect qualitative, fast answers. These four steps consist of:

1. Identify user's information needs– reflected users' insights of using unique identifiers, machine models and problem keywords
2. Identify data structure, data reliability and technical limitations through empirical investigation
3. Map data with user's information needs by providing data types to document fields in the Elasticsearch environment
4. Prioritise relevant data through Boolean querying

This strategy was devised and supported by measuring the first-use learnability, user satisfaction and relevance, which obtained positive results:

- first use learnability: 100% success for the task, 80% success without help and 27.93s average completion time
- user satisfaction: 5/5 positive comments
- relevance: 92%

The last sub-question ("What are potential challenges encountered in building a search portal MVP and how can those be tackled") exposed multiple problems that initially seemed entirely of a technical nature. Since the beginning, the search engine portal was on the verge of falling into the trap of too-much-disordered-data: multiple table data sources, nested type docs and chaotic email conversational data were some of the biggest challenges encountered in the discovery phase of the problem. These revealed a pattern that is present in information retrieval systems: dealing with the validity and reliability of current data. Potential solutions discovered were: understanding the relations between data sources and how they can still bring value to users. These insights appeared especially from using qualitative methods of research, such as contextual inquiries or interviews.

Another challenge was approaching the complexities of a full-stack solution, that was dealt with by performing a broad investigation of the technologies available and understand which can fit optimally the project requirements, both technical, cost-wise and related to the time frame imposed. What took the most time was figuring out the alignment of all the components to produce an effective query strategy. From here, the nature of the challenges took a humanistic turn: dealing with a vast number of features users desire and how those synchronize with the users' flow revealed that behind the development of a search engine tool lays a deep understanding of human psychology, context and behaviours. A non-judgemental mindset is the ethos of understanding behaviours, such as not framing people's actions as idiosyncrasies, but acknowledging them, even if they do not fit initial assumptions.

Lastly, an important aspect was managing users' time: it was essential to create awareness of the tool from the beginning, so users can assume responsibility and co-create the prototype with its builder. Their time needs to be booked at least one week in advance, and they must be involved regularly.

Contribution

The author's contribution lays on diagnosing the problem of information accessibility in a customer support unit. By creating a specialized searching tool, the author brings fast accessibility to a knowledge base that accumulated throughout the years but was not used at its full potential. The author tackles the problem not only from a pragmatic point of view – from the perspective of an information accessibility need - but also a hedonic one, showing concern about the users' ease to use

the tool, to facilitate the integration with their work pipeline. The author's solution opens the possibility for the stakeholders to integrate different knowledge bases mentioned in the project, thus creating a centralized tool for data retrieval. Additionally, the author proposes future steps to the challenges that have not yet been addressed due to the scope of the thesis.

References

[1] M. Alavi and D. Leidner, "Knowledge Management Systems: Issues, Challenges, and Benefits," *Communications of the Association for Information Systems,* vol. 1, pp. 3-4, 1999.

[2] D. W. Otter, J. R. Medina and J. Kalita, "A Survey of the Usages of Deep Learning in Natural Language Processing," *IEEE Transactions on neural networks and learning systems,* vol. 20, no. 10, 2019.

[3] B. Marr, "How much data do we create every day? The mind blowind stats everyone should read," *Forbes,* 21 5 2018.

[4] "Customer support and new product development - An exploratory study," *International Journal of Operations & Production Management,* vol. 21, no. 3, pp. 275-301, 2001.

[5] T. Peham, "Usersnap," 10 7 2019. [Online]. Available: https://usersnap.com/blog/email-customer-support-channel/. [Accessed 21 04 2020].

[6] E. Sawy and G. Bowles, "Redesigning the Customer Support Process for the Electronic Economy: Insights from Storage Dimensions," *MIS Quarterly,* vol. 21, pp. 457-483, 1997.

[7] M. Redbord, "The State of Customer Service in 2019," Hubspot, 2019.

[8] Z. Yan, "DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016.

[9] H. Kärkkäinen, "Managing customer information and knowledge with social media in business-to-business companies," in *11th International Conference*, 2011.

[10] J. Weizenbaum, "ELIZA—a computer program for the study of natural language communication between man and machine.," *Communications of the ACM,* vol. 9, no. 1, pp. 36-45, 1966.

[11] D. Benyon, P. Turner and S. Turner, Designing Interactive Systems: People, Activities, Contexts, Technologies, England: Pearson Education Limited, 2005.

[12] Y.-W. Liao, Y.-M. Huang, Z.-Y. Su and C.-W. Wei, "An Empirical Evaluation of Online Cooperative Programming Platforms Based on the PACT Framework and Technology Readiness," *Journal of Internet Technology,* vol. 20, no. 2, pp. 345-352, 2019.

[13] J. Reinius, *The PACT Analysis Framework, A case study of 1177.se,* Karlskrona: Blekinge Institute of Technology, 2011.

[14] K. Holtzblatt, J. B. Wendell and S. Wood, Rapid contextual design: A how-to guide to key techniques for user-centered design, San Francisco, CA: Morgan Kaufmann, 2005.

[15] N. Lal, S. Qamar and S. Shiwani, "Information Retrieval System and challenges with Dataspace," *International Journal of Computer Applications,* vol. 147, pp. 23-28, 2016.

[16] C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[17] J. Kalervo, "IR evaluations methods for retrieving highly relevant documents," *ACM SIGIR Forum,* vol. 51, no. 2, pp. 243-250, 2017.

[18] K. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal for Documentation,* vol. 28, no. 1, pp. 11-21, 1972.

[19] G. Salton, A. Wong and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM,* vol. 18, no. 11, pp. 613-620, 1975.

[20] S. E. Robertson, "Theories and Models in Information Retrieval," *Journal of Documentation,* vol. 33, no. 2, pp. 126-148, 1977.

[21] S. Robertson, C. Rijsbergen and M. Porter, "Probabilistic Models of Indexing and Searching," in *Proceedings of the 3rd Annual ACM Conference on Research and Development in Information Retrieval*, 1980.

[22] M. Pannu, A. James and R. Bird, "'A Comparison of Information Retrieval Models'," in *Western Canadian Conference*, 2014.

[23] Lv, Yuanhua and C. Zhai, "When documents are very long, BM25 fails!," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information*, 2011.

[24] R. A. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[25] D. M. W. Powers, New Methods in Language Processing and Computational Natural Language Learning, 1998.

[26] solid IT gmbh, "db-engines," 2020. [Online]. Available: https://db-engines.com/en/ranking_definition. [Accessed 22 7 2020].

[27] Splunk, "Splunk," Splunk, 2020. [Online]. Available: https://www.splunk.com/en_us/about-us/why-splunk.html. [Accessed 22 7 2020].

[28] D. I. Nikola Luburić, "Comparing Apache Solr and Elasticsearch search servers," in *6th International Conference on Information Society and Technology ICIST 2016*, Belgrade, Serbia, 2016.

[29] J. Blasenak, "Search Technologies," Accenture, 2020. [Online]. Available: https://www.searchtechnologies.com/blog/solr-elasticsearch-cognitive-search. [Accessed 21 7 2020].

[30] "Stack Overflow Uses Facets and Geo-Coding," Elastic, [Online]. Available: elastic.co/videos/stack-overflow-uses-facets-and-geo-coding. [Accessed 20 7 2020].

[31] "How Netflix is using Elasticsearch," Elastic, [Online]. Available: elastic.co/videos/netflix-using-elasticsearch. [Accessed 20 7 2020].

[32] "SoundCloud: Helping users find the sounds that move them," Elastic, [Online]. Available: elastic.co/customers/soundcloud. [Accessed 20 7 2020].

[33] M. Bajer, "Building an IoT Data Hub with Elasticsearch, Logstash and Kibana," in *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Prague, 2017.

[34] A. Brasetvik, "Elastic," Elastic, 16 9 2013. [Online]. Available: https://www.elastic.co/blog/found-elasticsearch-from-the-bottom-up. [Accessed 24 7 2020].

[35] B. Hundley, "Qbox," Qbox, 24 7 2015. [Online]. Available: https://qbox.io/blog/optimizing-elasticsearch-how-many-shards-per-index. [Accessed 24 7 2020].

[36] Opensource, "What is Docker?," Opensource, 2020. [Online]. Available: https://opensource.com/resources/what-docker.

[37] D. Wells, "Five Key Benefits of "Going Serverless"," Netlify, 6 8 2018. [Online]. Available: https://www.netlify.com/blog/2018/08/06/five-key-benefits-of-going-serverless/.

[38] C. Dahlqvist, "How many shards should I have in my Elasticsearch cluster?," Elastic, 18 09 2017. [Online]. Available: https://www.elastic.co/blog/how-many-shards-should-i-have-in-my-elasticsearch-cluster.

[39] J. Stoltz, "Search UI. Libraries for the fast development of modern, engaging search experiences.," Elastic, [Online]. Available: https://github.com/elastic/search-ui.

[40] Elastic, "Keyword datatype," Elasticsearch, [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/keyword.html.

[41] R. M. Jakob Nielsen, Usability Inspection Methods, New York: John Wiley & Sons, 1994.

[42] S. M. Z. Ahmed, C. McKnight and C. Oppenhein, "The use of a heuristic process to evaluate an online information retrieval interface.," *LIR,* vol. 30, no. 95, pp. 3-9, 2013.

[43] M. MAGUIRE, "Methods to support human-centred design," *International Journal of Human-Computer Studies,* vol. 55, no. 4, pp. 587-634, 2001.

[44] G. Pauvonic, "A usefull experience: Why usability is essential to ux design".

[45] H. Petrie and N. Bevan, "The evaluation of accesibility, usability and user experience," in *CRC Press*, 2009.

[46] R. Attar, T. Grossman and G. Fitzmaurice, "A survey of software learnability: Metrics, methodologies and guidelines," in *CHI 2009 Metrics*, 2009.

[47] M. P. Cota, S. Olschner, J. Thomaschewski, M. Rauschenberger and M. Schrepp, "Efficient measurement of the user experience of interactive products. How to use the user experience questionnaire," *International Journal of Artificial Intelligence and Interactive Multimedia,* 2013.

[48] A. Joyce, *How to Measure Learnability of a User Interface,* NN Group, 2019.

[49] H. H.Sayani, D. F.Rico and R. F.Field, "History of computers, electronic," *Advanced in Computers,* vol. 73, pp. 1-55, 2008.

[50] SAGE Publications, Inc., , "Triangulation," SAGE Publications, 2010.

[51] W. S. Robert B. Hanfield, "An assesment of manufacturing customer pain points: Challenges for researchers," *Supply Chain Forum: An International Jounal,* vol. 6, no. 2, pp. 6-15, 2005.

[52] J. Chisman, K. Diller and S. Walbridge, "Usability testing: A case study," *College Research Libraries,* vol. 60, no. 6, pp. 552-569, 1999.

[53] M. S. Andreasen, H. V. Nielsen and S. O. Schroder, "What happened to remote usability testing? an empirical study of three," in *Proceedings of the SIGCHI Conference on Human Factors in Computing*, New York, NY, USA, 2007.

[54] K. Moran and K. Pernice, *Remote moderated usability tests: How to do them.*

[55] M. McCloskey, "Turn User Goals into Task Scenarios for Usability Testing," 12 01 2014. [Online]. Available: https://www.nngroup.com/articles/task-scenarios-usability-testing/.

[56] G. Eysenbach and C. Köhler, "How do consumers search for and appraise," *BMJ,* vol. 324, no. 7337, pp. 573-577, 2002.

[57] A. Smyk, "Top UX Research Interview Questions to Ask Users," Adobe, 21 2 2020. [Online]. Available: https://xd.adobe.com/ideas/process/user-research/user-interview-questions-ux-research/.

[58] H. Jensen, "Don't Let Your Brain Deceive You: Avoiding Bias In Your UX Feedback," 12 10 2017. [Online]. Available: https://www.smashingmagazine.com/2017/10/avoid-bias-ux-feedback/#friendliness-bias.

[59] K. Moran, "Usability Testing 101," 1 12 2019. [Online]. Available: https://www.nngroup.com/articles/usability-testing-101/.

[60] McMaster University of Engineering, "Different_measures_for_evaluation," 24 10 2009. [Online]. Available: http://wiki.cas.mcmaster.ca/index.php/Different_measures_for_evaluation.

[61] D. Tamir, O. V. Komogortsev and C. J. Mueller, "An effort and time based measure of usability," *international workshop,* 2008.

[62] National Business Research Institute, "Measuring and Managing Customer Satisfaction," [Online]. Available: https://www.nbrii.com/customer-survey-white-papers/measuring-and-managing-customer-satisfaction/.

[63] B. Laugwitz, T. Held and M. Schrepp, "Construction an evaluation fo a user experience questionnaire," *Holzinger, A,* pp. 63-76, 2008.

[64] R. F. Dam and T. Y. Siand, "Interaction Design," 22 07 2020. [Online]. Available: https://www.interaction-design.org/literature/article/affinity-diagrams-learn-how-to-cluster-and-bundle-ideas-and-facts#:~:text=Affinity%20Diagrams%20can%20help%20you%20bundle%20and%20cluster%20large%20bodies,relations%20between%20groups%20of%20informatio.

[65] R. Budiu, "nngroup," 1 10 2017. [Online]. Available: https://www.nngroup.com/articles/quant-vs-qual/.

[66] J. Nielsen, Usability engineering, Morgan Kaufmann, 1994.

[67] J. Sauro, "What is a good task-completition rate?," MeasuringU, 21 3 2011. [Online]. Available: https://measuringu.com/task-completion/.

[68] T. Repke and R.Krestel, "Bringing Back Structure to Free Text Email Conversations with Recurrent Neural Networks," *Lecture Notes in Computer Science,* p. 114–126, 2018.

[69] P. Ragone, "Scaling Elasticsearch," 13 1 2017. [Online]. Available: https://medium.com/hipages-engineering/scaling-elasticsearch-b63fa400ee9e.

[70] M. Beseiso, "A New Architecture for Email Knowledge Extraction.," *nternational Journal of Web & Semantic Technology.,* vol. 3, 2012.

[71] B. Stecanella, "MonkeyLearn," 5 October 2017. [Online]. Available: https://monkeylearn.com/blog/analyzing-customer-support-interactions-on-twitter-with-machine-learning/.

[72] T. L. Keiningham, B. Cooil, T. W. Andreassen and L. Aksoy, "A Longitudinal Examination of Net Promoter and Firm Revenenue Growth," *Jounal of Marketing,* vol. 71, no. 0022-2729, pp. 39-51, 2007.

[73] "Stanford University," Stanford, 7 4 2009. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html. [Accessed 20 04 2020].

[74] M. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems,* vol. 14, pp. 130-137, 1980.

[75] M. Porter, "Snowball," City University, London, 2002. [Online]. Available: http://snowball.tartarus.org/.

[76] C. D. Paice., "Another Stemmer," *ACM SIGIR Forum,* vol. 24, no. doi: 10.1145/101306.101310., pp. 56-61, 1990.

[77] solid IT gmbh, "DB-engines," solid IT, [Online]. Available: https://db-engines.com/en/ranking_trend/search+engine,https://db-engines.com/en/ranking/search+engine. [Accessed 21 7 2020].

[78] M. . Bajer, "Embedded software development in research environment: A practical guide for non-experts," , 2014. [Online]. Available: http://ieeexplore.ieee.org/iel7/6851984/6862649/06862660.pdf?arnumber=6862660. [Accessed 21 7 2020].

[79] J. Nielsen, "Thinking Aloud: The #1 Usability Tool," Nielsen Norman Group, 15 01 2012. [Online]. Available: https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/.

[80] M. Rosala, "How to Analyze Qualitative Data From UX Research: Thematic Analysis," 29 9 2019. [Online]. Available: https://www.nngroup.com/articles/thematic-analysis/.

[81] J. Nielsen, "Nielsen Norman Group," 1 11 1994. [Online]. Available: https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/.

[82] S. K. Arora, "what-is-data-analysis-methods-techniques-tools," Hackr.io, 14 5 2020. [Online]. Available: https://hackr.io/blog/what-is-data-analysis-methods-techniques-tools.

[83] T. T. Quan, S. C. Hui and A. C. M. Fong, "Automatic Fuzzy Ontology Generation for Semantic Help-Desk Support," *IEEE Trans. Ind. Inf.,* vol. 2, no. 3, pp. 155-164, 2006.

[84] T. L. Acorn and S. H. Walden., "SMART: Support management automated reasoning technology for Compaq customer service.," in *Proceedings of the fourth conference on Innovative applications of artificial intelligence.*, 1992.

[85] D. Kelly, D. J. Harper and B. Landau, "Questionnaire mode effects in interactive information retrieval experiments," *Information Processing & Management,* vol. 44, no. 1, p. 122–141, 2008.

[86] C. Pautasso, E. Wilde and R. Alarcon, REST: Advanced Research Topics and Practical Applications, New York: Springer-Verlag, 2014.

[87] F. Reichheld, "Harvard Business Review," December 2003. [Online]. Available: https://hbr.org/2003/12/the-one-number-you-need-to-grow. [Accessed 6 April 2020].

List of Figures

List of Equations

List of Tables

Annexe 1

```
{
    "infomogen": {
        "mappings": {
            "properties": {
                "Contact": {"type": "text","fields": {"keyword": {"type": "keyword","ignore_above": 256}}
                },
                "DeliveryDateField": {"type": "date","format": "dd/MM/yyy"},
                "ProblemKeyword1": {"type": "keyword" },
                "ProblemKeyword2": {"type": "keyword"},
                "EquipmentFrecuency": {"type": "keyword",},
                "EquipmentModel": {"type": "keyword",},
                "Mails": {"type":"nested","properties": {
                    "Body": {"type": "text"},
                    "Cc": {"type": "text","fields": {"keyword": {"type": "keyword","ignore_above": 256}}},
                    "From": {"type": "text","fields": {"keyword": {"type": "keyword","ignore_above": 256}}},
                    "Mail": {"type": "keyword",},
                    "Subject": {"type": "text","fields": {"keyword": {"type": "keyword","ignore_above": 256}}},
                    "Title": {"type": "text","fields": {"keyword": {"type": "keyword","ignore_above": 256}}},
                    "To": {"type": "keyword",}}},
                "EquipmentMounting": {"type": "keyword",},
                "OpeningDateField": {"type": "date","format": "dd/MM/yyy"},
                "EquipmentID": {"type": "keyword",},
                "CaseName": {"type": "text","fields": {"keyword": {"type": "keyword","ignore_above": 256}}},
                "EquipmentID": {"type": "keyword",},
                "ShortDescription": {"type": "text","fields": {"keyword": {"type": "keyword","ignore_above": 256}}},
                "EquipmentVelocity": {"type": "keyword",},
                "CaseKeyword": {"type": "keyword",},
                "EquipmentVoltage": {"type": "keyword",},
                "OrderID": {"type": "keyword",}
                }
            }
        }
    }
}
```

Annexe 2

```
{
  highlight: {
      fragment_size: 200,
      number_of_fragments: 1,
      fields: {
        ShortDescription: {},

      }
    },
  _source: ["CaseId", "ShortDescription","EquipmentModel","CaseName","CaseKeyword","ProblemKeyword1"
  ,"EquipmentID","OpeningDate","DeliveryDate", "Mails",""],
    aggs: {
      Type1: { terms: { field: "CaseKeyword.keyword", size: 30 } },
      FaultCode1: { terms: { field: "ProblemKeyword1.keyword", size: 50 } },
      MachineType: { terms: { field: "EquipmentModel.keyword", size: 300 } },
    },
    query: {
      bool: {
        must: [? {
                multi_match: {
                query: searchTerm,
                fields: ["EquipmentName","ProblemKeyword1","CaseID","EquipmentID", "OrderID"]
        }
      }
    : { match_all: {} };],
        ...(filter && { filter })
      }
    },
    ...(sort && { sort }),
    ...(size && { size }),
    ...(from && { from })
  };

  return body;
}
```

Annexe 3

```yaml
version: '3'

services:
  elasticsearch: # Elasticsearch Instance
    container_name: SearchEng
    image: docker.elastic.co/elasticsearch/elasticsearch:7.9.0
    volumes: # Persist ES data in seperate "esdata" volume
      - esdata:/usr/share/elasticsearch/data
    environment:
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - discovery.type=single-node
    ports: # Expose Elasticsearch ports
      - "9300:9300"
      - "9200:9200"

volumes: # Define seperate volume for Elasticsearch data
  esdata:
```

Annexe 4

	QUALITATIVE
	Comments
User 1	The proccess went well, the main challenge was to figure out what the filter represented without previous explanation, first 1, second 3, third 2. The positive experience was to see how easy fast I have access to the emais data, also being able to filter by problem code feels very good to take out irrelevant data.
User 2	The process was easy, no challenges were found. First:3, Second:2 ,Third:1. The overall search engine feels good
User 3	Process was not difficult, I found everthing very clear, first:3 second:2 third:1, the filtering system was very useful but it should be more useful to have a specific filter only for the machine model and another for the machine specification code
User 4	Process was easy, I found only one thing confusing and is that the list of emails was very long so sorting them by date of received would be good first:3 second:2 third:1. The filtering is good but it should have a general button to undo all changes
User 5	It was very easy, I did not have any challenge and the user interface is good,first: 2, second 3, third 1. My positive experience from the process is that in general the is clear.