Improving customer support efficiency through decision support powered by machine learning

Simon Boman

Supervisor: Valency Colaco
Examiner: Simin Nadjm-Tehrani

External supervisor: Teodor Ganestål
Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Simon Boman
Abstract

More and more aspects of today’s healthcare are becoming integrated with medical technology and dependent on medical IT systems, which consequently puts stricter requirements on the companies delivering these solutions. As a result, companies delivering medical technology solutions need to spend a lot of resources maintaining high-quality, responsive customer support.

In this report, possible ways of increasing customer support efficiency using machine learning and NLP is examined at Sectra, a medical technology company. This is done through a qualitative case study, where empirical data collection methods are used to elicit requirements and find ways of adding decision support. Next, a prototype is built featuring a ticket recommendation system powered by GPT-3 and based on 65 000 available support tickets, which is integrated with the customer supports workflow. Lastly, this is evaluated by having six end users test the prototype for five weeks, followed by a qualitative evaluation consisting of interviews, and a quantitative measurement of the user-perceived usability of the proposed prototype.

The results show some support that machine learning can be used to create decision support in a customer support context, as six out of six test users believed that their long-term efficiency could improve using the prototype in terms of reducing the average ticket resolution time. However, one out of the six test users expressed some skepticism towards the relevance of the recommendations generated by the system, indicating that improvements to the model must be made. The study also indicates that the use of state-of-the-art NLP models for semantic textual similarity can possibly outperform keyword searches.
Acknowledgments

First of all, I would like to thank Sectra for giving me the opportunity to conduct this thesis on their behalf. I would like to thank my supervisor Teodor Ganestål and my entire team at Sectra for valuable feedback, help, and support throughout my thesis. I would also like to thank everyone at Sectra’s support division who participated and helped me during my study, as well as everyone else who has been involved or shown interest in my work. Lastly, I would like to thank my examiner Simin Nadjm-Tehrani and my supervisor Valency Colaco for providing me with feedback on my work throughout the thesis.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstract</strong></td>
<td>iii</td>
</tr>
<tr>
<td><strong>Acknowledgments</strong></td>
<td>iv</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>v</td>
</tr>
<tr>
<td><strong>List of Figures</strong></td>
<td>viii</td>
</tr>
<tr>
<td><strong>List of Tables</strong></td>
<td>x</td>
</tr>
</tbody>
</table>

## 1 Introduction
1.1 Motivation ........................................... 1  
1.2 Background ........................................... 2  
1.3 Aim .................................................. 2  
1.4 Research Question ................................. 2  
1.5 Approach ............................................. 2  
1.6 Delimitations ....................................... 3  

## 2 Related Work
2.1 Methods for Data Collection and Requirements Elicitation ........................................... 4  
2.1.1 Requirements Engineering and Requirements Elicitation ........................................... 4  
2.1.2 Interviews ........................................... 5  
2.1.3 Participant Observations .......................... 6  
2.1.4 Qualitative Data Generation and Analysis ........................................... 7  
2.2 Machine Learning and Natural Language Processing ........................................... 8  
2.2.1 Machine Learning .................................... 8  
2.2.2 Natural Language Processing for Unstructured Textual Data .................................... 8  
2.2.3 Natural Language Processing for Semantic Text Similarity .................................. 8  
2.2.4 Generative Pre-trained Transformer 3 ........................................... 10  
2.2.5 Feature Selection .................................... 10  
2.2.6 Data Cleaning ....................................... 11  
2.3 Decision Support ....................................... 12  
2.3.1 Decision Support in Ticketing Systems using a Recommendation System ................. 13  
2.3.2 Machine Learning in Decision Support ........................................... 13  
2.4 Methods for Evaluation ........................................... 14  
2.4.1 Methods for Evaluating Decision Support ........................................... 14  
2.4.2 Methods for Evaluating Recommendation Systems ........................................... 15  
2.4.3 Qualitative Evaluation and Qualitative Case Studies ........................................... 16  
2.4.4 User Perceived System Usability and Satisfaction ........................................... 16  
2.4.5 Member Checking .................................... 17  

## 3 Method
3.1 Prestudy and Requirements Elicitation ........................................... 18
# Table of Contents

1. **Introduction**
   - 1.1 Background and Research Questions
   - 1.2 Methodology
   - 1.3 Contribution

2. **Literature Review**
   - 2.1 Previous Work
   - 2.2 Related Work

3. **Methodology**
   - 3.1 Interviews
     - 3.1.1 Interview with the Dynamics Service Owner
     - 3.1.2 Observational session with the First Line Support
     - 3.1.3 Interviews with the First Line Support
     - 3.1.4 Observational session with the Second Line Support
     - 3.1.5 Interviews with the Second Line Support
     - 3.1.6 Interview with the Head of Scandinavian Support
     - 3.1.7 Member Checking to validate elicited requirements
   - 3.2 Implementation of a Prototype Recommendation System
     - 3.2.1 Training Data Acquisition and Initial Feature Selection
     - 3.2.2 Data Preprocessing and Cleaning
     - 3.2.3 Evaluation & Selection of Machine Learning Models
     - 3.2.4 Final Feature Selection and Hyperparameter tuning
     - 3.2.5 Deployment & Integration of Prototype
   - 3.3 Evaluation
     - 3.3.1 End User Evaluation
     - 3.3.2 Qualitative Evaluation with Interviews
     - 3.3.3 Quantitative Evaluation using System Usability Scale

4. **Results**
   - 4.1 Prestudy and Requirements Elicitation
     - 4.1.1 Findings from the interview with the Dynamics Service Owner
     - 4.1.2 Findings from First Line Support observations
     - 4.1.3 Findings from First Line Support interviews
     - 4.1.4 Findings from Second Line Support observations
     - 4.1.5 Findings from Second Line Support interviews
     - 4.1.6 Findings from the interview with the Head of Scandinavian Support
     - 4.1.7 Member checking to decide on ticket recommender feature
   - 4.2 Implementation of a Prototype Recommendation System
     - 4.2.1 Training Data Acquisition and Initial Feature Selection
     - 4.2.2 Data Preprocessing and Cleaning
     - 4.2.3 GPT-3 Model
     - 4.2.4 Final Feature Selection and Hyperparameter Tuning
     - 4.2.5 Pipeline for Generating Recommendations
     - 4.2.6 Prototype Used for the Evaluation
   - 4.3 Evaluation
     - 4.3.1 Qualitative Evaluation with Interviews
     - 4.3.2 Quantitative Evaluation using System Usability Scale

5. **Discussion**
   - 5.1 Results
     - 5.1.1 Results from Interviews
     - 5.1.2 Results from System Usability Scale (SUS) Evaluation
   - 5.2 Method
     - 5.2.1 Implementation of the Recommendation System
     - 5.2.2 Evaluation of the Prototype
     - 5.2.3 Replicability, Reliability, and Validity
   - 5.3 Ethical and Societal Aspects

6. **Conclusion**
   - 6.1 Future Work

7. **Bibliography**

A  **Interview Questions for the Service Owner Interview**
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Overview of the final support ticket pipeline</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>Screenshot of the prototype used for the evaluation</td>
<td>35</td>
</tr>
<tr>
<td>4.3</td>
<td>Responses to question 6 - Has the quality of your work improved, remained the same, or declined since you started using the recommendations?</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>Responses to question 7 - From a scale of 1 to 5, how accurate do you believe the recommendations provided by the system are?</td>
<td>37</td>
</tr>
<tr>
<td>4.5</td>
<td>Responses to question 8 - Do you trust that the recommendation system is giving you relevant recommendations? Yes/No?</td>
<td>37</td>
</tr>
<tr>
<td>4.6</td>
<td>Responses to question 9 - When you perceive a recommendation as not relevant, do you believe this is because the system failed to find the most similar tickets, or because tickets of a similar nature simply do not exist?</td>
<td>38</td>
</tr>
<tr>
<td>4.7</td>
<td>Responses to question 2 - In what ways have the recommendations improved (or worsened) your efficiency?</td>
<td>38</td>
</tr>
<tr>
<td>4.8</td>
<td>Responses to question 14 - How does using the recommendations compare to other methods you've used to find similar tickets, e.g. keyword search?</td>
<td>38</td>
</tr>
<tr>
<td>4.9</td>
<td>System Usability Scale score for each of the test users and the average SUS score.</td>
<td>39</td>
</tr>
<tr>
<td>4.10</td>
<td>Average System Usability Scale score for each of the statements.</td>
<td>39</td>
</tr>
</tbody>
</table>

### C.1 Responses to question 1 - Is there any functionality or aspect of Dynamics you think works poorly, takes an unnecessarily long time to perform, or leads to mistakes in your work? | 58   |

### C.2 Responses to question 3 - If you could decide on new functionality to be added in Dynamics, what would you want? | 59   |

### D.1 Responses to question 1 - Is there any functionality or aspect of Dynamics you think works poorly, takes an unnecessarily long time to perform, or leads to mistakes in your work? | 60   |

### D.2 Answers and their frequencies to question 2 - If you could decide on new functionality to be added to Dynamics, what would you want? | 61   |

### G.1 Responses to question 11 - Do you believe that your efficiency working with tickets has increased from using the recommendations? Yes/No? | 71   |

### G.2 Responses to question 2 - In what ways have the recommendations improved (or worsened) your efficiency? | 71   |

### G.3 Responses to question 3 - Do you believe that you need to spend less time searching for similar tickets and thus find the solution to a problem faster when using the recommendations? Yes/No? | 72   |

### G.4 Responses to question 4 - Do you believe that the recommendations have helped you find relevant cases which you would otherwise not have found manually? Yes/No? | 72   |

### G.5 Responses to question 9 - Do you believe that the difficulty of performing your daily work has decreased using the prototype? Yes/No? | 72   |

### G.6 Responses to question 6 - Has the quality of your work improved, remained the same, or declined since you started using the recommendations? | 72   |
<table>
<thead>
<tr>
<th>Question</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.7</td>
<td>From a scale of 1 to 5, how accurate do you believe the recommendations provided by the system are?</td>
</tr>
<tr>
<td>G.8</td>
<td>Do you trust that the recommendation system is giving you relevant recommendations? Yes/No?</td>
</tr>
<tr>
<td>G.9</td>
<td>When you perceive a recommendation as not relevant, do you believe this is because the system failed to find the most similar tickets, or because tickets of a similar nature simply do not exist?</td>
</tr>
<tr>
<td>G.10</td>
<td>How happy are you with the benefits the recommendations provide as they are today?</td>
</tr>
<tr>
<td>G.11</td>
<td>Would you want to continue using it in the future if it were integrated in Dynamics?</td>
</tr>
<tr>
<td>G.12</td>
<td>How do you believe the recommendations will impact the efficiency of the entire support division?</td>
</tr>
<tr>
<td>G.13</td>
<td>In what other contexts do you believe the recommendations would be most useful?</td>
</tr>
<tr>
<td>G.14</td>
<td>How does using the recommendations compare to other methods you’ve used to find similar tickets, e.g. keyword search?</td>
</tr>
<tr>
<td>G.15</td>
<td>Is there something you think is working poorly, or anything you wish would have been done differently or changed?</td>
</tr>
<tr>
<td>G.16</td>
<td>Would you like this prototyped feature to be implemented in Dynamics?</td>
</tr>
<tr>
<td>G.17</td>
<td>Do you have any feedback or suggestions for improving the recommendation system, or anything else to comment on?</td>
</tr>
</tbody>
</table>
List of Tables

2.1 The four observational types ............................... 7
2.2 The five possible ratings and their mapping to the System Usability Scale score ............................... 17
1 Introduction

This chapter aims to introduce the reader to the underlying problem addressed in the thesis and the research area of this study, along with relevant background information. The thesis is conducted in collaboration with Sectra Medical, a Swedish medical technology company headquartered in Linköping, Sweden.

1.1 Motivation

As the population of the world grows, and the average human lifespan is increasing in both developed and non-developed countries, more and more people are requiring medical care. This results in a huge strain on healthcare all around the world, and apart from medical advances, technical advances within the field of medical technology are also needed in order to streamline the healthcare process. Because of this, medical technology companies are playing an increasingly important role in healthcare.

Companies delivering medical technology solutions have customers primarily consisting of hospitals and various other medical institutions. Since the systems used in healthcare are life-critical, where any downtime in the absolute worst-case scenario could result in the death of a patient, the aspects of availability and reliability are of paramount importance. Thus, it becomes imperative that these companies maintain high-quality, responsive customer support that can quickly provide assistance to ensure that critical systems are live and functioning correctly.

Due to the aforementioned reasons, a considerable amount of resources must be spent on the availability of customer support, particularly by companies delivering medical technology systems. One of the many ways to enhance and increase the efficiency of customer support would be to implement decision support for the support personnel, as this could result in less time being spent on average per ticket. This would have the benefit of efficiency improvements to healthcare as healthcare employees can receive help resolving their issues faster, which could lead to better care for patients and decreased strain on existing resources. Additional benefits from the faster response times include cost savings and improved customer satisfaction. These benefits are not exclusive to Sectra, and can be generalized to any company that provides customer support services.

One possible way to add decision support in order to increase the efficiency of support personnel is by facilitating difficult and time-consuming tasks, which can be done with the
help of machine learning. In the case of textual data such as support tickets containing natural language, Natural Language Processing models can be utilized.

1.2 Background

One of the leading companies within the medical technology industry is Sectra Medical, which is the company where this study will be conducted. Sectra is headquartered in Linköping and was founded in 1978 with its roots in research conducted at Linköping University. Sectra’s main division, Sectra Medical, develops and delivers primarily medical imaging systems. An example of such a system is PACS (Picture Archiving and Communications System), which is a medical imaging technology utilized by most modern hospitals. The PACS stores and handles medical images from various medical diagnostic procedures such as ultrasounds or MRIs, and enables convenient storage and access to the system from multiple subsystems.

For handling support tickets, Sectra’s customer support division uses Microsoft’s CRM (Customer Relationship Management) system Dynamics 365 hereinafter referred to as Dynamics. For the majority of received support tickets, a similar issue has often been handled previously by someone on the support team. Consequently, a typical first step in the workflow of dealing with a new ticket is to search for similar tickets amongst previously solved cases (closed tickets) with the goal of finding a solution. This process is manual and can be very time-consuming as tens of thousands of tickets are available, but once a similar case has been found it often provides a good starting point which reduces the overall resolution time. Additionally, textual inconsistencies due to different documentation standards, different languages, spelling mistakes, and use of synonyms further complicate the search.

Even without textual inconsistencies, keyword searches can be problematic. Keyword searches can be ambiguous, and thus it can be difficult for a user to obtain the desired search results [66]. Due to the ambiguity of keyword searches the user is often forced to reformulate their query multiple times until the sought-after results are retrieved [9].

1.3 Aim

The underlying aim of this thesis is to investigate and deliver insight into how state-of-the-art machine learning and natural language processing models such as GPT-3 can be used for semantic textual similarity. This will be examined through a case study at the customer support division at Sectra, with the goal of increasing customer support efficiency.

1.4 Research Question

In order to achieve the above aim, the following research question will be studied:

- How can machine learning be used to implement decision support, and how does this affect customer support efficiency?

1.5 Approach

To fulfill the above aim and answer the research question, a new enhanced version of the current support system in use at Sectra will be developed and evaluated qualitatively using the current system as a baseline. The project will be divided into a prestudy phase, an implementation phase, and an evaluation phase.

In the prestudy phase, a literature study will be conducted that encompasses relevant topics from the research area. In addition to this, requirements will be elicited from the support at Sectra, regarding what could be done to increase the efficiency of the customer support
division. The requirements will be elicited through qualitative empirical research methodologies, such as empirical observations and interviews with different groups of end users in the support division.

In the implementation phase, natural language processing models and their use in similar contexts will be studied through a literature review, and the most suitable architecture capable of effectively handling unstructured textual data will be selected. Following this, a prototype will be developed to fulfill the requirements elicited during the prestudy.

In the final evaluation phase, the impact of the prototype system on customer support efficiency will be evaluated. The evaluation will be conducted by having a subset of the users the requirements were elicited from use the prototype in their daily work for five weeks. The efficiency will be measured in terms of the support personnel’s perceived efficiency. This will be measured qualitatively through interviews and emphasis will be placed on how the end user perceives the decision support and its effects on their daily work. In addition, a quantitative System Usability Scale score will be used to measure the user-perceived usability and satisfaction of the prototype.

Ultimately, the criteria for success is that the proposed prototype system can increase the efficiency of the support personnel in their daily work. The goal is to increase efficiency by reducing the average resolution time of a support ticket, and thus efficiency will hereinafter be considered in terms of this during the study.

1.6 Delimitations

As this thesis is conducted in collaboration with Sectra, some technical restrictions are imposed. Making changes to the production code affecting the live support system may not be feasible solely for the evaluation of the prototype, and this may thus require workarounds. Additionally, as the study is conducted using data provided by Sectra, there is no way to evaluate this work in another domain on a different dataset to replicate the findings. Due to time constraints, five weeks are planned for six end users to test and evaluate the prototype, which limits the amount of data generated. Due to the limited amounts of data from the evaluation, a quantitative evaluation involving metrics such as average resolution time will not be feasible. With limited amounts of data and due to the varying complexities of support tickets, it would be difficult to isolate the impact of the prototype from other aspects affecting resolution time, and too much randomness would be introduced.
2 Related Work

This chapter aims to present relevant related work which the methodology for the data collection and evaluation will build upon. The chapter will also exhibit related work on research related to natural language processing and decision support which will inspire and be used during the implementation.

2.1 Methods for Data Collection and Requirements Elicitation

In the following section, the methods that are used for data collection and requirements elicitation are presented. The presented methodology will be used during the prestudy phase of the study.

2.1.1 Requirements Engineering and Requirements Elicitation

An important part of developing a system is the requirements engineering, which relies heavily on the initial phase of requirements elicitation [18]. Requirements are normally elicited from various stakeholders through interviews and other empirical methods. Relevant stakeholders are identified depending on the use case and target end user of the system in mind.

In a paper on requirements engineering techniques by Macaulay, various suitable methodologies for requirements elicitation is presented [39]. The author argues that typical methods well suited for eliciting requirements are interviews, questionnaires, and observational studies. Furthermore, Macaulay discusses common failures when working with requirements elicitation, and presents two major findings. The author argues that the two most frequent points are communication problems and lack of knowledge or understanding, which both stem from miscommunication between the person eliciting the requirements and the end users. With this, Macaulay highlights the importance of good communication, particularly in the early stages of requirements elicitation.

In a study on requirements engineering, Coughlan and Macredie put further emphasis on the importance of communication in regard to requirements elicitation [16]. The authors argue that the elicitation and communication of user requirements are highly important parts of system development, but also highly error-prone parts of the development cycle. Additionally, they argue that communication during requirements elicitation plays a vital role in the success of the entire project. This is due to the fact that this stage often involves people with
2.1. Methods for Data Collection and Requirements Elicitation

widely different positions in the organization, backgrounds, and knowledge, amplifying the risks of poor communication and potential misunderstandings.

The authors suggest that when eliciting requirements for a project with a vague problem definition, a human-centered approach with close collaboration and communication with the end users is required. In order to successfully do so, an understanding of the user and their environment must be established. A proposed way to gain this understanding is to perform on-site observations of the users, their needs, and how they operate in their natural environment, both in the system but also in the organization as a whole. Thus, under circumstances when dealing with a vaguer goal, the authors propose initiating the elicitation upon an observational case study or a similar empirical form of data collection.

In a survey and evaluation of various requirement elicitation techniques for computer-based systems, Goguen and Linde point to research showing that many projects fail due to improper requirements elicitation [20]. Further on they argue that inadequate requirements lead to two typical types of project failures, the first one where no such system can be built that satisfies all the requirements due to technical constraints or other types of limitations, and the second where the final requirements are in practice not what is needed. Thus, the authors suggest that enough time is spent eliciting requirements in order to avoid such failures, in particular to avoid the second type of failure which is typically the most common case. Lastly, several different data collection methods are presented, all of which the authors suggest as suitable requirements elicitation methods. Among those are interviews, questionnaires, and participant observation methods where the researcher attempts to become a part of the community in the environment of interest.

2.1.2 Interviews

When it comes to the use in requirements engineering, interviewing is a common data collection method, and a study on requirement engineering practices by Kassab shows that interviewing is the most common requirements elicitation technique [28]. The author analyses two studies which received 93 respectively 247 respondents from people involved in the requirements elicitation phase. In both surveys, interviews were a part of the requirements elicitation process in more than 50% of all cases. In another study by Ferrari et al. the authors present evidence that question-based elicitation methods, such as interviews, are both common and very useful in practice [18].

In a paper by Runeson and Höst, various guidelines for conducting case study research for software engineering are presented [55]. They argue that case studies, which are characterized by investigating a phenomenon in its natural context, are suitable for research within software engineering, due to the possibility of studying a system, human behavior, and their interaction in its natural context. The authors mean that one of the major research methodologies for data collection from a specific population is interviews. Gathering information through interviews is a direct method, meaning that the data is collected directly from the subjects in real-time. Although direct methods are generally more expensive in terms of time for both the subject and researcher (compared to indirect methods such as questionnaires), they bring benefits due to the extent of control which the researcher has and thus can decide on specifically what data to collect, how to collect it, and in what context. The authors suggest initiating interviews by stating the purpose of the research, as well as how the data will be handled and used. Additionally, asking some simple questions related to the interviewee’s background and role is also recommended to establish a connection. After the initial introduction, the interview continues to the main set of interview questions.

Runeson and Höst categorize interview questions into two types, open and closed. Open questions are broader and can be questions such as What do you like the most about the current system? which invite for a longer discussion on a certain topic, whereas closed questions are
2.1. Methods for Data Collection and Requirements Elicitation

more limited, and could be as straightforward as a yes/no question. Additionally, the authors define interviews to be of one of the following three types: unstructured, semi-structured, or structured. In a structured interview, all questions and their order are planned before the interview, and there will be no deviations from the plan in terms of questions and order. On the contrary, an unstructured interview is not as planned but loosely based on certain general concerns or interests, and the interview will develop from those as the interview unfolds. Semi-structured interviews are a combination of both, as a list of questions to be answered is prepared, but their ordering is not set. Additionally, the semi-structured interview allows for follow-up questions and opens up for discussion. Typically, semi-structured interviews are the most common type, and they often consist of both open and closed questions.

Lastly, the authors present three different types of principles of how the interview can be conducted, the funnel model, the pyramid model, and the hourglass model. The different models represent different ways to structure the open and closed questions for the interview, assuming the set contains both open and closed questions which is typically the case. The funnel model starts out with open questions and ends with closed questions, whereas the pyramid model does the exact opposite and starts out with closed questions and finishes off with open questions. The hourglass model on the other hand has open questions at the start and end of the interview, leaving the closed questions to be asked in the middle of the interview.

2.1.3 Participant Observations

Participant observation is, according to Seaman in a paper on the topic, where the researcher collects data by observing the subject in its natural environment [60]. Seaman claims that the main purpose of performing participant observations is to observe the actions and behavior of the participants, some of which would potentially otherwise not be discovered.

A large benefit of observations is that they can be useful when there is an apparent or suspected deviation from the 'official' view of a phenomenon that is being observed. E.g. the view from managers or decision makers, compared to the view from the users that work closely with the phenomenon that is being observed. [54]. Additionally, due to the direct examination occurring in real-time in the natural context, observational methods may provide insight regarding topics the participants would otherwise be unaware of, unable to think of, or simply not wish to talk about [41]. However, observational methods can also be challenging due to the so-called Hawthorne effect, which is when participants change their behavior, intentionally or unintentionally, due to being under observation, which can result in the observer receiving an untrue perception.

Seamen continues by suggesting a standard way of data collection through observations [60]. This should be done by taking so-called field notes, which are brief notes aimed to be used as support to document as much detail as possible as soon as the observational session has ended. The field notes should be thorough enough to allow for proper documentation, and contain information like context, participant, and of course important events and other comments from the researcher. However, the level of detail strongly depends on the aims of the study, and a study of a more exploratory kind often requires more detail, compared to a study with a narrow scope. After the observational session, the field notes can then be refined and any findings more thoroughly documented.

A broader view of observations and how they can be conducted is discussed by Runeson and Höst, who claim that observations can be conducted in many different ways, with the common factor being the investigation of how certain tasks are conducted [55]. In general, they differentiate between two different ways of conducting participant observation, which they argue can be done either in a direct manner together with the participant(s) or indirectly. In the more direct way, the researcher can take on a more engaging role with the participants. This form of observation typically comes off as unnatural for many people, but can help
discover a lot of tacit knowledge or unconscious behavior. On the contrary, in the indirect way, the researcher takes on a more passive role and simply observes the participants or phenomena in their natural context.

Based upon the degree of interaction, as well as the degree of awareness of being observed, Runeson and Höst divide empirical observations into the following four types presented in table 2.1 [55].

<table>
<thead>
<tr>
<th>High interaction</th>
<th>Low interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>Type 3</td>
</tr>
<tr>
<td>Type 2</td>
<td>Type 4</td>
</tr>
</tbody>
</table>

Table 2.1: The four observational types.

Observations belonging to types 1 and 2 where there is a high degree of interaction between the researcher and participants are common when the researcher is a part of the team which is being observed, and thus is not simply seen as a researcher without any connection to the team. For types 3 and 4, where there is a lower degree of interaction, the researcher is more seen as just a researcher. As these four settings will yield different results from the observational study, the authors suggest that one should use the same type or combination of types and stick with it throughout the study.

2.1.4 Qualitative Data Generation and Analysis

When working with qualitative data collection, a popular way to formalize the gathered data into a set of requirements, statements, or discoveries is to use some sort of theory generation method [60]. These type of methods build upon using gathered qualitative data, such as field notes or answers to interview questions, to narrow down and refine the most common findings into points that provide insight and deeply describes the phenomena being researched.

A proposed method for this by Seaman is constant comparison, which builds upon constantly aggregating notes, comments, interview answers, or other qualitative data into more concise statements [60]. These statements are then reexamined and compared when additional data is collected in order to define common aspects of the phenomena. When it comes to aggregating the gathered data, this can be done by either preformed or postformed categorization. Using preformed categorization, a list of statements which are to be explored is predefined, and once the data has been gathered the findings are distributed to the most relevant statement on the list. With postformed categorization the opposite is done, firstly the data is collected, followed by aggregating and grouping the statements together. Aggregating the field notes using postform categorization is typically the most suitable way when the objectives of the study are initially open and not focused on a particular aspect. Once all data has been generated, the last step is to extract and document the most relevant discoveries, which can range from a list of bullet points to longer documentation of complex propositions.

As a proposed additional step, the authors suggest trying to provide more evidence to back this up by strengthening the findings. A typical way to strengthen any finding based upon qualitative methods, is to ensure the validity of the methods that were used to generate it. For empirical studies consisting of e.g. observations and interviews a good way is by making sure that the observees and interviewees, as well as the context, are representative of the phenomenon that is being studied. Another proposed way to strengthen the findings is triangulation, which is ensuring that you have evidence supporting your proposition collected from multiple sources (e.g. different persons), or collected and generated using different methodologies.
2.2 Machine Learning and Natural Language Processing

The purpose of the following section is to present a theoretical background on natural language processing (NLP) as well as different NLP architectures and models.

2.2.1 Machine Learning

The field of machine learning has been under heavy development during the past years, not only within research but also within practical applications [42]. One of the most popular areas of research, and also one of the areas within machine learning with the highest practical applicability, is natural language processing (NLP) [25] [30]. Natural Language Processing is a subfield of linguistics and machine learning, which builds upon using various computational algorithms and techniques to enable a computer to understand, learn, and generate natural human language [25].

2.2.2 Natural Language Processing for Unstructured Textual Data

The primary application of NLP is for processing natural language generated by humans and represented as textual data for various purposes [30]. An example is an NLP model that takes a text as input, and returns the most similar text amongst other texts in a database. Two trivial low-level tasks within NLP are tokenization and lemmatization. Tokenization refers to splitting up a sentence into multiple shorter tokens, where the delimiter between tokens typically is any blank space, but longer words can be split into and represented as multiple shorter tokens. The second important basic step is lemmatization, where a word's (or tokens') suffix is removed in order to convert the word to its root form. Additional problem-specific steps are also applied, but these two steps are the very foundation of most NLP models and are fundamental for any type of further processing.

There are various application areas, such as categorization and clustering of texts, and these types of NLP solutions have previously been used for the categorization of support tickets or complaint requests to streamline work and make sure that the ticket or request is directly sent to the correct person [30].

In a recent study by Li et al. the usage of NLP for unstructured textual data in electronic health records is studied [34]. This type of data is presented as unstructured and free-form text written by healthcare workers, which results in difficulties in further processing the data. Thus, the authors argue that models built on top of classic machine learning algorithms like SVM (Support Vector Machines) or kNN (k-nearest neighbors) have difficulties performing well in most tasks due to the unstructured and domain-specific textual data. The authors argue that more complex models, such as deep neural network-based architectures, will yield better results for these types of scenarios.

2.2.3 Natural Language Processing for Semantic Text Similarity

Text similarity has for a long time been a popular area of research as it can be applied in a wide range of scenarios and use cases, and with recent advances, the field of semantic text similarity has also made great advances [27]. The recent major developments have mainly been enabled through recent advances within AI and machine learning, but also due to increased computational efficiency and the availability of larger corpuses of natural language text, all of which combined allow for creating and training more sophisticated models [14].

The use cases of NLP for text similarity can range from measuring textual similarities among entire documents such as patents [61], to finding similarities among rules or procedures across documents [52].

Text similarity is an important part of many NLP tasks, such as finding and retrieving information, translation, and generating recommendations, among others [52] [14]. Bag-of-Words,
Term Frequency or Inverse Term Frequency are traditional methods used for text similarity, and have proven to be useful together with text search and e.g. querying by keywords \cite{14}. However, these types of algorithms simply split a text into tokens and calculate the frequency of each token, and based upon these similarities are computed. Thus, these types of models can not be used when the detection of semantic similarities is particularly important. By instead using so-called word embeddings, a model can learn to understand syntactic similarities better. The creation of word embeddings is typically done by the vectorization of text, i.e. by transforming and representing a text as a numerical vector in a multi-dimensional vector space. With embedding models, deep learning is typically used to learn how to do the vectorization by focusing the training on the concept that words that appear in a syntactically similar context, often share semantic similarities. A huge breakthrough within the research on word embeddings was the introduction of the Word2vec algorithm which is trained using deep neural networks\cite{38}. The algorithm uses large corpuses to build a vocabulary and learns the vector representation for each word in the vocabulary, which enables similarity to be measured between two texts using e.g. the cosine similarity.

When using word embeddings, as the texts are converted into and represented as numerical vectors, the cosine similarity can then be used to retrieve the similarity between two vectors in the same vector space by computing the cosine angle between the two vectors \cite{52}. Thus, two proportional vectors will have a cosine similarity of 1 indicating the highest possible semantic similarity, and two inverse vectors will have a cosine similarity of -1 indicating the lowest possible semantic similarity \cite{45}. As the training of the embedding model is based upon the fact that semantically close words often appear in contexts that are syntactically similar, embeddings of semantically similar texts, sentences or words will be close in the vector space, and thus the cosine similarity can be used to measure their semantic similarity. Different similarity metrics for measuring text similarity exist, but using the cosine similarity is typically the preferred way of measuring similarity when semantics are important, as other similarity measurements, such as the Jaccard similarity, are based upon matching individual characters or words and thus not able to capture semantic similarities \cite{52}.

In a comparative study on text similarity using vector space models, term frequency models such as TFIDF and BOW are compared to topic models (a statistical approach where texts or documents are assigned a probability distributing to how similar it is to certain pre-defined topics), as well as neural network models such as text embedding models like Word2vec and Doc2vec \cite{61}. The authors conclude that when semantic meaning is important, neural network models outperform both topic models and term frequency models primarily due to their ability to learn the semantic meaning and also take the ordering of words into account. They pose that these neural network models can be either trained by following the assumption that semantically similar words will appear in a similar context, or by having the model learn by feeding it pairs of semantically similar words until the model can properly evaluate semantic meaning.

From a survey on semantic text similarity, various models are benchmarked and compared over a dataset consisting of various popular datasets used for comparing models for semantic similarity \cite{14}. The authors focused on three distinct types of models, knowledge-based methods, corpus-based methods, and deep neural network-based methods, as well as hybrids between these. Among the deep neural network models, CNNs, LSTMs, Bi-LSTMs, and transformer-based models were compared.

The survey found that deep neural network-based models outperform all other types of models, in particular when it comes to capturing semantic similarity. A major reason for this is that the complex architectures of these models can benefit largely from increased amounts of data and computational power. The authors argue that the downside of using the superior deep neural network-based models is the demanding computational resources needed
to both train and use these types of models, thus in certain contexts the tradeoff between improved performance and drastically higher computational costs should be considered. They also mean that in highly domain-specific use cases, the data from which the models were pre-training on may differentiate from the domain-specific data to such an extent where it is difficult to produce rational results, even with fine-tuning. Additionally, typically deep learning models are so-called black-box models, meaning that the results they produce are difficult to interpret.

The authors also conclude that among the deep neural network models, transformer-based models such as XLNET, T5-11B, and BERT outperform all other deep neural network models at this task and achieve state-of-the-art results. Transformer models are designed to capture the semantic meaning of words in embeddings which builds upon the concept of self-attention (i.e. depending on the context learn which bits of data are more important than others). These models are built and trained upon an architecture split into two parts, an encoder part and a decoder part, where the encoder layers encode which parts of the input data that are related, followed by the decoder doing the opposite thing using the encodings as input.

### 2.2.4 Generative Pre-trained Transformer 3

GPT-3 (Generative Pre-trained Transformer 3) is one of the most popular state-of-the-art NLP models for most NLP tasks built upon the transformer architecture [51]. GPT-3 is often compared to and seen as the successor to previous transformer-based models such as XLNET and BERT. GPT-3 was released in 2020 by OpenAI and uses the same model and architecture as their previous model GPT-2, and its 175 billion parameters were trained using tokens primarily generated through web crawling [19]. The model is trained using contextual word embeddings, meaning that the semantics of tokens is different in different contexts, and thus the representation of a token is dependent upon the rest of the text [51]. At the release of GPT-3, the model had more than 10 times the amount of parameters than any previous non-sparse language model, and compared to other fine-tuned models GPT-3 achieved state-of-the-art performance in certain tasks even without any fine-tuning [11]. GPT-3 was pre-trained in such a way that it would perform well on multiple tasks, which was a different approach from other NLP models which typically were designed for a specific task only. GPT can thus be used off-the-shelf with its parameters frozen and deliver good performance [51]. However, if required parameters may also be unfrozen in order to fine-tune the model (tuning it to be more suitable to your domain and context by additional training of the model on available data from that domain and context).

There are numerous application areas and types of tasks where GPT-3 can be utilized and which are publicly available [44]. Numerous base models are available tailored toward specific tasks, such as text similarity. For the task of semantic text similarity using an embedding model is according to GPT-3’s documentation optimal, which is also in line with previous literature study. Among the available embedding models, text-embedding-ada-002 is the latest and current state-of-the-art model. Using this model also supports measuring similarity using cosine similarity, which previously was concluded to be the optimal similarity metric.

GPT-3 has achieved state-of-the-art performance at various tasks, beating other transformer-based models such as various BERT (Bidirectional Encoder Representations from Transformers) variants [4]. In a recent comparison from 2022 for knowledge-based question answering, the model based upon GPT-3 achieved an accuracy 48.0% compared to the previous state-of-the-art at 39.4% accuracy [65].

### 2.2.5 Feature Selection

In machine learning tasks feature selection plays a vital role in a model’s success, as the selected features, the number of total features, and their quality will have an impact on the
2.2. Machine Learning and Natural Language Processing

The feature selection is typically the first step of any data preprocessing, and is particularly useful when the data is high-dimensional \cite{56}. More often than not, data from real-world applications contains both redundant and noisy features. Performing feature selection will yield a simpler model that is easier to understand and which performs better for its intended task, as irrelevant features are ignored, and redundant features are handled, thus reducing risks of overfitting. Another benefit of reducing the dimensionality is the reduced costs both in terms of decreased storage and increased computational efficiency. This dimensionality reduction can be performed in two ways, either via feature extraction where the features are projected to a lower dimensionality feature space, or by selecting features as they are and ignoring the rest. The latter is usually the preferred method, especially when the features are interpretable and understandable to a human, it may be beneficial to keep them as they are to better enable interpretability and analysis of the model. Although, it is important to keep in mind that selecting too few features will also result in poor performance, as some relevant information may be eliminated.

For many classical predictive machine learning problems, there are suitable models which can be utilized that perform automatic feature selection, and thus the most relevant features for a model can be automatically obtained \cite{35}. However, when working with NLP tasks and textual data, these types of automatic feature selection methods are often not suitable. This is particularly the case when your data consists of just one or a few strings. In practice it has been shown that automatic feature selection is outperformed by manual feature selection performed by domain experts \cite{3,2}. Manual feature selection by someone with domain knowledge, albeit usually much more time-consuming, will ensure that a relevant set of features are chosen. A recommended approach for feature selection is to first remove features containing irrelevant data for the model, as well as redundant features \cite{43}. This step is then often followed by having a domain expert determine which remaining features are relevant to keep and which are not.

2.2.6 Data Cleaning

As the quality of data from real-life applications typically is of low quality and contains a lot of unwanted noise, data cleaning is a very common preprocessing step used for machine learning and similar fields \cite{64}. The goal of data cleaning is ultimately to improve the performance of the model, as noisier data leads to worse performance or an unusable model. In addition to this, certain statistical models can be heavily affected by the prevalence of heavy outliers, thus depending on the context this may also be something that is cleaned from available training data.

However, depending on the type of data, the detection as well as salvaging of dirty data may be difficult, but failing to do so can heavily penalize the performance of the model \cite{15}. This phenomenon gave birth to the popular concept of “Garbage in, garbage out”, which refers to the fact that feeding any model bad data as input will result in the output being bad as well. Data cleaning can be performed in both a qualitative where perceived noise is removed, or quantitatively by using e.g. statistical methods like outlier detection. Common examples of dirty data typically applicable in most domains are things such as missing values, redundant information, inconsistent formats, or values breaking business rules. The cleaning itself must consist of two sequential phases, first a detection phase, followed by the restoration phase. Both of these phases pose the question regarding what type of dirty data to detect/repair, how to detect/repair it, and lastly where to detect/repair it. As you typically discover more flaws in your data the more you clean it, it is recommended to work iteratively when dealing with data sets where this may be true such as textual data, and iterate between detecting and repairing \cite{36}. For numerical data, the detection and repair can sometimes be done using statistical approaches, e.g. automatic outlier detection, whereas for textual data a
Decision Support

The cleaning of data is often done as a step in exploratory data analysis, which is the iterative process of exploring data by visualizing, followed by cleaning and feature selection, in order to draw insights from the data and better prepare it for modeling. Compared to performing visualization, feature selection and cleaning one time sequentially, by performing these steps iteratively you can maximize your insight into the data. It can also enable the resulting data cleaning to be of higher quality, as there are more chances to detect anomalies and outliers. A popular tool for exploratory data analysis is using histograms to visualize the data, as this is an easy and efficient way to gain information about a distribution as well as detect and remove noise and outliers.

The authors of a paper on semantic text similarity argue that the importance of preprocessing and cleaning holds extra true when working with textual data, and particularly unstructured textual data, as failing to do so will yield poor results when computing similarities using e.g. the Jaccard similarity or Cosine similarity. The writers also argue that when working with unstructured text and NLP, a common approach is to use Regular Expressions for detecting and cleaning certain patterns from a text, as this is the most frequent, and most punishing, type of dirty data encountered when working with textual data.

Decision Support

The following section exhibits theory on decision support, as well as a case study on how a recommendation system can be used to create decision support in a ticketing system.

Human decision-making is a fundamental part of our lives, and is consequently a popular research topic, in particular how to in an effective manner support decision making. There is a wide range of semi-structured decision problems which typically have some predefined parameters which may not be changed, but still require human input on some other parameters. Typical examples of decision-making problems are managerial business decisions, where some parameters are present and fixed, whereas some are not, and these unknown parameters thus rely on human input. These types of semi-structured decision problems are common in all types of domains, and subsequently, Decision Support Systems and various subtypes emerged during the 1960s to support decision-makers in various settings.

Decision Support Systems (DSS) can be seen as a subset area of the more general Information Systems (IS) area, with the focus on presenting information to aid in or improve decision-making. Today, Decision Support Systems are prevalent in all types of organizations and multiple subtypes of DSSs exist, where most companies employ some form of integrated decision support in another IS. Traditionally, DSSs were primarily used as an aid for managerial decisions. Although still popular to use for managerial questions, they are now used in a broader scope. The goal of decision support is that when a human is working with an IT system to solve a problem, the system should help by automating the provision of useful information. Such a system is not meant to solve the target problem on its own, which in many cases is also not feasible, but simply to increase the effectiveness of the human using the system by eliminating various tedious and time-consuming tasks such as information retrieval. Thus, in practice, a DSS is less of an actual technology and more of a philosophy on how to develop certain information systems.

In a research paper by Shim et al. decision support technology is discussed, and how such a system can be developed. The authors suggest that the biggest emphasis is on the model development and the problem analysis. The authors mean that in a typical case, the first step should always be the identification and analysis of the problem to be solved, followed by developing and implementing the most suitable models to help solve the problem.
2.3.1 Decision Support in Ticketing Systems using a Recommendation System

The use of decision support in ticket systems is explored in a case study by Revina and Rizun, where the goal is to implement a recommendation system based upon unstructured text data to enable decision support in the ticketing system [53]. A recommendation system is a system providing recommendations typically based upon the similarity of the object of interest, thus helping users find relevant objects to examine [22]. The generation of recommendations is performed in two steps. First, the similarities between the object of interest and other candidates are computed, followed by recommending the most suitable objects to support the end user’s decision. Revina and Rizun argue that recommendation systems can provide great opportunities, in particular when it comes to time savings [53]. The authors propose that the best way to do so is by using a knowledge-based recommendation system.

A knowledge-based recommender system builds upon acquiring knowledge from people with knowledge of the domain in which the system will operate, often by reasoning what requirements must be met in order to meet the user’s needs [12]. An example of a knowledge-based recommender system is a system based upon a set of rules or constraints [17]. In such a system, a set of rules or constraints must be fulfilled in order to be considered as a recommendation. The purpose is typically to be able to impact what types of recommendations are feasible and also wanted. Thus, these constraints are typically user-defined and extracted from domain experts.

In the ticketing system case study by Revina and Rizun [53], tickets about related problems are processed using the same template, but searching using keywords performs poorly when trying to find a template. Due to this poor search functionality, preexisting templates are easily overlooked due to the flaws in the system. The lack of a well-functioning search function in this application domain results in inefficient work and increased time spent per ticket. The proposed recommendation system from the study thus has the goal to, given the text of a ticket, provide help to the user in finding the best solution for the problem or request in the ticket. The authors suggest different types of solutions to handle different types of complexities. In the case of a high-complexity ticket, the suggested solution is to simply provide the user with previous similar tickets. Concisely, the entire ticket handling cycle starts with the preprocessing of ticket text, followed by the prediction and computation of recommendations of similar tickets, and lastly, these recommendations are provided to the end user who is processing the ticket which the recommendations were based upon.

The evaluation of the results from the case study was conducted qualitatively, where 13 users were interviewed and reviewed the recommendation system. The interviews followed a semi-structured funnel approach and had a focus on the practical feasibility and applicability of the recommendation system. Ultimately, the authors claim that based upon the final qualitative evaluations, the recommender system and the historical ticket recommendations received positive feedback from the 13 users, indicating that such a system would increase the effectiveness of their work and lead to time savings.

2.3.2 Machine Learning in Decision Support

The area of machine learning can be applied to decision support, creating so-called Intelligent Decision Support Systems (IDSS) [7] [40]. The goal is that by integrating machine learning into a traditional DSS to create a more intelligent system able to provide stronger decision support [57]. According to a survey on decision support, one of the most common types of Decision Support Systems are IDSS (Intelligent DSS) [7]. Due to advances in AI and machine learning, various AI technology has been proven to be useful in facilitating human decision-making, especially when dealing with real-time decision problems [57]. In general, the integration of AI technology is mostly useful in a complex environment, where a regu-
lar DSS is not enough. IDSSs can also typically handle ambiguity better, and also deal with completely new situations or data in a better manner than the more rigid traditional DSSs.

A common technique to utilize in IDSSs is the use of Artificial Neural Networks (NN), which is a black box (meaning that it is not possible to interpret what happens in the model between the input and output) machine learning model [57] [40]. A Neural network consists of many processing units called neurons which are interconnected in layers in a way that mimics the human brain, hence the name. Neural Networks differ from other AI methods that are sequential and logic-based, as for Neural network there does not have to be a specific relationship between the inputs and output, as a fundamental advantage of Neural Networks is that they due to their structure naturally embed the representation of nonlinearity. Thus, for complex tasks, Neural Networks can provide great advantages over other models that are logic-based.

2.4 Methods for Evaluation

Below, methodologies for evaluation are presented. This chapter will focus on covering the qualitative evaluation of decision support and recommendation systems in a case study, as well as a quantitative metric for measuring user-perceived usability and satisfaction. This methodology will be used for evaluating the outcome of the study during the evaluation phase.

2.4.1 Methods for Evaluating Decision Support

A fundamental way of understanding the benefit of enhancing a system with decision support is by evaluating the decision support it provides [49]. For Decision Support Systems, and in particular intelligent ones, Wren suggests that the improvement can be measured in multiple ways [57]. Typically the primary way of evaluation is measuring the user’s increased efficiency. Ultimately, this increase in efficiency is often translated into terms of reduced costs or increased revenue, as such a tangible improvement is easily comprehended. Wren suggests that this improvement can be measured both qualitatively (e.g. by self-rating or expert rating), and quantitatively (e.g. by measuring and comparing the number of decisions in a fixed time interval).

Another view on evaluating decision support is to, depending on the type of decision support, look directly at the process of the decision making or the outcome from the decision making [49]. When evaluating by looking at the process of working with the integrated decision support, qualitative metrics are often used, such as improved efficiency, an increased degree of systematic work, speed, or perceived better understanding of the problem from the user. Comparably, when evaluating the outcome of the decision support quantitative measurements are usually used, such as financial measurements or the accuracy of decision support (accuracy in terms of how many times the decision support was useful compared to how many times it was not).

The empirical evaluation of decision support is examined in a paper by Adelman [5]. Adelman claims that due to the complexity and unstructuredness of problems which Decision Support Systems often handle, they also become difficult to evaluate empirically. Because of this Adelman, like many others, argue that the empirical evaluation of Decision Support Systems is often lacking, both in practice and in previous research. The authors continue by stating that evaluation, both in terms of verification (making sure that the system is modeled and programmed correctly), and validation (examining if the system does indeed solve the problem and is of use to the end-user), where the latter is more often overlooked, is extremely important. A suggested way to perform the evaluation of the system, given that the system can be deployed to its intended environment, is to evaluate the system in real-time on new
2.4. Methods for Evaluation

Data as this way the data is also new to the person doing the evaluation, thus reducing any chance of introducing bias. Another proposed way is to evaluate the decision support by having an independent set of domain experts test the performance by integrating the decision support with their daily work, rather than relying on a single person performing a strictly theoretical evaluation.

In another paper on empirical evaluation for decision support systems by Adelman, various ways for evaluating decision support are presented and compared [6]. The author supports the claim that in the industry the evaluation of decision support systems is often lacking, which should be a vital part in order to determine the usefulness of the implemented decision support. The author means that evaluation should not only be conducted after the system has been developed and deployed, but should rather be an integrated part of the entire cycle of design and development of the system in order for the system to be consistent with the stakeholder’s requirements. The author argues that the evaluation focus should not lie on the software quality or the processes of the development which is often the focus when developing a traditional software system (although these aspects are of importance, they are not directly tied to the decision support aspect), but rather lie on the evaluation of the decision support from the end users view.

Further on, the author examines different empirical evaluation methods, and suggests the use of case studies for evaluating decision support. The author suggests that in early stages, such as when experimenting with proof-of-concepts, experimental case studies where the prototype is being used are often feasible methods for the evaluation of the proposed system. These types of experiments are typically conducted by having participants perform various representative tasks, which then allow for objective measures such as the quality of the decision, as well as more subjective measures such as the perception and confidence of the user when making the decision. However, in later stages where the system has been deployed and is operating in its target environment, the author suggests that the better alternative would be to conduct shorter case studies. Compared to a more experiment-like case study, at this stage, a case study has fewer independent variables that may be varied, as the decision support system is investigated while operating in its practical context. The author means that case studies are a popular way of evaluating software systems, in particular management information systems, and thus they are also usable when evaluating decision support systems. This type of empirical testing, in addition to receiving user input regarding their satisfaction with the system, also allows for testing the accuracy of the system in a real context and on real data.

2.4.2 Methods for Evaluating Recommendation Systems

For the evaluation of a recommendation system, just as the end-user perception is the focus when evaluating decision support, the same approach is often taken when evaluating a recommender system [48]. In a paper on the evaluation of recommender systems, Knijnenburg et al. propose using a user-centric evaluation [31]. The authors argue that strictly quantitative metrics such as accuracy do not necessarily correlate to high satisfaction or usability of a system, in particular when the recommendations generated are fed to a human end user. Instead, the authors suggest using a user test approach where users test the system “live” and base the evaluation upon this, as opposed to “offline” algorithm evaluation. The authors propose gathering subjective opinions via interviews or questionnaires, with Likert scale questions indexed from “I totally disagree” to “I totally agree”, in order to measure the perceived quality of the recommendations. Together with evaluating the recommendations themselves, the authors also propose measuring the user experience in order to evaluate the perceived effectiveness of the recommendation system.
Herlocker et al. argue that when using a recommendation system with the purpose of creating decision support, the evaluation focus should be on the user and the way they perform tasks in the system under evaluation [23]. The authors mean that these two aspects are often intertwined, as although a recommendation system can generate and feed the user with recommendations, this is nothing more than decision support as ultimately the user has to make a decision, whether it is something received from the recommendation system or not. The authors illustrate this with a movie recommendation site, where a user can be given a candidate set of similar films given a specific title, which provides the user with decision support albeit the user still has to make the decision to select any of the candidates. Based on this, the authors argue the importance of focusing on the user’s experience and user-perceived benefits when evaluating the recommendations of a recommender system.

Pu et al. also suggest measuring a user’s perceived accuracy, where the accuracy refers to whether the user is confident and trusts that the recommendations are accurate or not [50]. The authors argue that this is also an important aspect to complement when evaluating from a user-centric perspective. A proposed way is using a Likert scale ranging from "strongly disagree" to "strongly agree" to measure this.

2.4.3 Qualitative Evaluation and Qualitative Case Studies

Case studies, in particular qualitative ones, are a popular form of evaluation across many industries [63], as has also been mentioned in the above sections. In a paper on the topic by Wilson, the author argues that in terms of qualitative evaluation, case studies can produce informative evaluations [63]. In another paper on the topic, Paprini et al. argue that case studies can provide an in-depth exploration of a phenomenon in its natural contextual environment, thus enabling better understanding and insight into a problem [46]. Additionally, qualitative data is generally richer than quantitative data, as the amount of information that can be retained is higher than for quantitative methods [60]. This makes qualitative data work well for evaluating certain types of systems where the end user is in focus, or phenomena where the research perspective is broad.

According to Runeson and Höst in a paper about case study research, the authors compare various methodologies based on the study’s primary objective, primary data, and design [55]. They classify case studies as having an exploratory objective (finding new insights and ideas in the area of the case study), primarily having qualitative data, and the design to be flexible (parameters and assumptions can be changed throughout the study). The authors continue by arguing that for case studies and due to the nature of their design, results are often analyzed in a qualitative manner in order to arrive at a conclusion. This is further supported by Seaman in a paper on qualitative methodologies in empirical studies, who suggests that both qualitative data collection methodologies, as well as qualitative methodologies for analyzing and drawing conclusions from the generated data, are suitable in a case study context [60]. The authors suggested ways of collecting qualitative data in empirical studies are typical empirical research methodologies, such as interviews, focus groups, and observations.

This was also supported in the above section in the case study using qualitative interviews to evaluate the decision support provided by a ticket recommender system in [53].

2.4.4 User Perceived System Usability and Satisfaction

When it comes to user acceptance testing of any type of IT system or prototype, usability testing is one of the most popular and feasible ways of testing and a good way to get an overall perceived quality of the system [33]. A typical way of measuring usability and user-perceived effectiveness or satisfaction of a system is by utilizing the System Usability Scale (SUS), which is a standardized 5-point Likert scale questionnaire indexed from “strongly agree” to “strongly disagree” with 10 statements designed to assess the perceived usability of a system and user satisfaction [33] [10]. A study from 2009 showed that SUS testing was a
part of the evaluation in 43% of all system usability evaluations [59]. It is recommended that the SUS form is filled out after the test users have had enough time to use and experience the system being evaluated [33].

After a test user has answered the 10 statements, the score which is ranging from 0 to 100 is computed [33]. The individual score for each statement is computed by taking the 5-point scale position minus one for odd rows, or five minus the 5-point scale position for even rows. The final SUS score is calculated by taking the summation of the score for each statement and multiplying this by 2.5.

As the result of each SUS evaluation is a score ranging from 0 to 100, Lewis proposes that in order to concretize the score take the average SUS score from all test users to obtain an overall SUS score. In addition to this, he proposed a curved grading scale mapping the SUS score to a grading of the SUS score based upon 446 studies which had used SUS as a part of their evaluation. In a similar survey based on 2324 SUS surveys, the mean SUS score was 70.14 [8]. The grading based on the SUS-score from that survey is presented in table 2.2.

<table>
<thead>
<tr>
<th>SUS-score</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 78.9</td>
<td>A</td>
</tr>
<tr>
<td>72.6 - 78.8</td>
<td>B</td>
</tr>
<tr>
<td>62.7 - 72.5</td>
<td>C</td>
</tr>
<tr>
<td>51.7 - 62.6</td>
<td>D</td>
</tr>
<tr>
<td>&lt; 51.6</td>
<td>F</td>
</tr>
</tbody>
</table>

Table 2.2: The five possible ratings and their mapping to the System Usability Scale score.

2.4.5 Member Checking

During the evaluation stage of a study, Seaman proposes so-called member checking as a strong way of strengthening your findings [60]. The method is based on getting feedback on your finalized findings from the same subjects from which you initially collected the data. By doing so, you can strongly strengthen your propositions by receiving support from the subjects regarding the relevance and quality of your findings which conclusions may be based upon. This is particularly important in cases where the conclusions from the study may affect the way the subjects perform their work duties.

Member checking is also suggested by Lincoln and Guba in a book on qualitative research methods [37]. The authors argue that member checks can also be a strong way to test both data and interpretations in an early stage of the study, as well as for testing the conclusions and findings in the later stages. Apart from performing checks with the participants from which the data was collected, the authors also suggest that this can be done together with other relevant stakeholders. The authors state that member checking can be either formal or informal. They argue that informal checks occur continuously throughout any type of data collection or analysis, as the researcher continuously tries to verify that the data and interpretations made are correct. More formal member checking is typically conducted towards the end of the study and has the intended purpose of claiming the credibility of the findings by verifying the findings with previous subjects and/or relevant stakeholders. A proposed way for performing more formal checking is to plan interview sessions together with members from each source group and relevant stakeholders and elicit data to strengthen the findings and propositions as well as gather criticism.
3 Method

The following chapter serves the purpose of presenting and describing the methodology used for the study. The study was divided into three major parts, and below methodology for each of the three parts is presented, based upon the methodologies and related works presented in chapter 2.

The project was structured into the following three major parts, Pre-study and feature elicitation, Implementation, and Evaluation.

The first phase of the project was the pre-study and feature elicitation. The main focus of the pre-study was a literature review consisting of reading and studying papers on relevant methodology and related work for the study. After the pre-study, the gathered theories and methodologies were used in order to elicit requirements through various empirical software engineering principles as discussed in section 2.1. Once requirements were selected and engineered, the project moved on to the implementation phase where a recommendation system prototype was developed. Once the prototype was deployed, the end results were evaluated. During the evaluation of the end results, the criteria for success was that the proposed prototype system can increase the efficiency of the support personnel in their daily work, which was evaluated primarily qualitatively as discussed in section 2.4.

In order to find relevant research papers to base the work upon, a method for finding and reading academic papers by Keshav was utilized [29]. The technique illustrates that you should read every paper in multiple passes. The goal of the first pass is to spend 5-10 minutes in order to get a general overview of the paper, by first reading the title, abstract, and introduction, followed by reading the titles of sections and sub-sections, and lastly the paper’s conclusions. Based upon this first pass, you decide whether you want to continue to do a second pass if you find the paper to be of interest and its correctness and clarity are valid. For the second pass, the goal is to spend an hour or more reading the paper more carefully in order to get a good understanding of the paper’s contents, as well as get a better grasp of the paper’s correctness and trustworthiness.

3.1 Prestudy and Requirements Elicitation

The requirements elicitation consisted of interviews, observations, and member checking. Interviews were conducted together with the Dynamics service owner to discover other relevant stakeholders and to gain a grasp of the domain. After this, observations of the support
3.1. Prestudy and Requirements Elicitation

personnel were conducted in accordance with subsection 2.1.3 in order to get familiar with the system used and an understanding of the workflow. Once familiarized with the system and workflow, interviews were prepared and conducted with selected employees according to the interview methodology presented in subsection 2.1.2. Finally, the gathered information from the interviews was used for member checking to finalize the final list of requirements to implement. As was discussed in subsection 2.1.1, ensuring good communication and understanding of the domain and its problems are extremely important at this stage of the study and crucial to the entire study, thus a large emphasis was put on this throughout the requirements elicitation.

3.1.1 Interview with the Dynamics Service Owner

In order to get introduced to the support division and the system in use, as well as technical possibilities and limitations in regards to the study and the evaluation of it, an interview was held with the service owner of the Dynamics service desk. The goal of the interview was primarily to examine what type of new features could create business value, and to identify relevant stakeholders to elicit requirements from as part of the first phase of the project. The interview followed a semi-structured approach, where certain questions about various topics of interest were planned, but room for discussing new topics was included. The questions were mainly open, but a few closed questions were also prepared. The interview followed the funnel model, where the interview started out with broader open questions and narrowed down towards closed questions. The interview was initiated by explaining the role of the researcher and the purpose of the study, as well as establishing some mutual background information. The list of questions prepared for the interview can be viewed in appendix A.

3.1.2 Observational session with the First Line Support

In order to gain a deeper knowledge of Dynamics, how the support personnel work with it, and their overall workflow, an observational session together with the first line support engineers was conducted for a full work day. Observations with four first line employees were conducted, all of which had the role of First Line Support Engineer. Three of them had been working in the role for roughly six months, and the fourth had roughly two years of experience in the role. The observation was based on the methodology described in subsection 2.1.3 and the observation was commenced by a brief introduction of the researcher and an explanation and motivation of the study and this particular observation. The purpose was to collect as much information as possible about how the support engineers interact with Dynamics, and which aspects of their work take up unnecessary time, are hard to do, or are error-prone. The goal was to discover findings which would not be able to be found through traditional interviews. In accordance with the theory, as the workflow of the support engineer can be easily visually tracked and followed, observations are suitable in this type of environment. In accordance with the proposed observational methodology presented, notes were taken throughout the observation session, in order to support proper documentation after the session has ended. After the observation, these notes were refined into more concise statements using a postform aggregation of the field notes. The observations alternated between low and high interactions with the participants, but stay fixated at a high awareness of observation, thus resulting in a mixture of a type 1 and type 3 observation as presented in section 2.1.3.

3.1.3 Interviews with the First Line Support

To elicit formal requirements from the first line support engineers, interviews were prepared after the observation and conducted on the same day. The questions were based upon the information that was gathered during the observation. The interview followed a structured
3.1. Prestudy and Requirements Elicitation

3.1.1 Approach with pre-defined questions, and the questions were all open. The interview was initiated by explaining the role of the researcher and the purpose of the study and the data collection of the current interviews in particular.

Below follows the topics which were discussed during the interviews:

1. Which aspects of Dynamics do you believe work poorly, take an unnecessarily long time to perform, or lead to mistakes in your work?
2. If you could decide on new functionality to be added to Dynamics, what would you want?

3.1.4 Observational session with the Second Line Support

To gain an even deeper knowledge of how the support personnel work with Dynamics, an observational session was also planned with the second line support for a full workday. Three employees were part of the observations, and all had the role of Application Specialist working as a Second Line Support Engineer. All three of them had been working for 10 or more years in other parts of the company before starting as an Application Specialist. They had worked in the current role for 1 year, 3 years, and 4 years respectively.

As the second line support deals with tickets escalated by the first line support, they typically deal with more complex issues which require deeper technical troubleshooting. The goal was to gather more information on how support engineers interact with Dynamics from another perspective to complement the findings from the first observation. The methodology and motivation for this observational session are identical to the first observation described in 3.1.2. The second line support consists of multiple smaller teams typically specialized in a specific product or product area. The observation was conducted with the Application Specialist team, whose primary task is to make sure that the customers’ applications are maintained and working correctly, as well as troubleshooting issues which may occur. The Application Specialist second line support team was deemed to be the most suitable team for the second observation, as their expertise area is broad with both deep technical knowledge and product knowledge. Additionally, they interact with most of the other teams they have a good understanding of how the other support teams work with Dynamics as well. Based upon this, they were deemed to provide the best representation of the entire second line support.

3.1.5 Interviews with the Second Line Support

Similarly to what was conducted with the first line support, interviews for eliciting more formal requirements from the second line support engineers were planned and held after the observation session. The interview followed a structured approach with pre-defined questions, and the questions were all open. The interview was initiated by explaining the role of the researcher and the purpose of the study and the data collection of the current interviews in particular. The exact same discussion topics as the ones for the interviews with the first line support were posed.

3.1.6 Interview with the Head of Scandinavian Support

To confirm the findings and get some second opinions from a managerial perspective, a brief interview was held with the head of the support division. This also served as a sort of member checking, which does not necessarily need to be conducted with the same participants from which the data was collected from, but can also be done with relevant stakeholders. The goal was to examine whether the findings were relevant and feasible to implement, as well as options for evaluation during the study. The interview followed an unstructured approach with no pre-defined questions, and the questions were open.
The first topic of discussion was regarding the relevance of the findings, followed by a discussion regarding the feasibility of integrating a prototype with the new feature into the support’s workflow for the evaluation of the work.

3.1.7 Member Checking to validate elicited requirements

After the gathering of thoughts and requirements from various stakeholders, member checking was conducted with the first and second line support engineers. The first and second line support engineers were selected for the member checking as they are the group most likely to be affected by any implemented changes in Dynamics as it would affect their daily work more than other groups. As the outcomes of the project may affect the way the support personnel conducts their daily work, member checking was in this case of large importance. The goal was to strengthen the findings and the proposed final feature before implementation by receiving support from the same participants from which it was elicited from.

3.2 Implementation of a Prototype Recommendation System

The second phase was the implementation of the prototype recommendation system. The first step was to build a data pipeline from Dynamics where new support tickets are imported to Azure which was the computational platform used for the project. This was followed by finding and evaluating appropriate machine learning models and algorithms for finding and generating recommended cases. Lastly, a prototype tool was developed and deployed and integrated with the current system in use, where the recommendations could be easily displayed and accessed given a new support ticket in Dynamics.

3.2.1 Training Data Acquisition and Initial Feature Selection

The first step was to acquire the training data which feature selection, data preprocessing, and cleaning was based upon. This training data consisted of all previously solved customer support tickets stored in the Dynamics database. The data was obtained by directly querying the Dynamics 365 database using the OData v4 API. As a first step of the pipeline to be built later, the support ticket data was directly read into Microsoft Azure which is the computational platform which was used throughout the project. This corresponded to roughly 104 000 tickets from the past five years. Once all the training data had been obtained and stored, an initial feature selection where only features which may have been useful for the modeling were kept. In accordance with what was discussed in subsection 2.2.5, manual feature selection was used. This was done iteratively by removing features containing irrelevant or redundant information until a set of useful features for the modeling were left.

3.2.2 Data Preprocessing and Cleaning

The next step after initial features were selected from the data was to preprocess and clean the data. This was done in accordance with what was presented in subsection 2.2.6, where the detection and cleaning of noise was done iteratively using an exploratory data analysis approach. The cleaning primarily relied on qualitative detection and removal of noisy patterns, but also quantitative in order to detect outliers when analyzing the length of tickets or various fields.

The tickets and their selected 8 features kept for modeling were iteratively visualized, analyzed, and cleaned. To be consistent with the language, the first step was to use a language detection algorithm from the Python package Spacy to detect any non-English paragraphs and translate them into English. This was followed by iterating visualizing and analyzing a sample of 500 tickets by using the package Pandas as well as by printing the tickets contents in order to detect patterns to clean. The detection and removal of noisy or irrelevant
data from the support tickets was done primarily using regular expressions and the Python package Re was utilized. Cleaning was done for sensitive data such as personally identifiable information (which should by default be anonymized by the first line support) as well as customer information. Thus, the patterns for detection and cleaning for this type of data were personal identity numbers, names, exam IDs, phone numbers, e-mail addresses, and customer information such as names or addresses.

Apart from sensitive information, cleaning was done for noisy data. The support ticket data, in particular some features, were initially very dirty, thus this was the area where most of the cleaning was done. The data contained a lot of long error messages, generic logs, or other copy-pasted information, which made most tickets very hard to follow initially. In addition to this, cleaning was done for systematic noise present in many tickets, such as common phrases and support template phrases, noise from tickets created via e-mail, noise included from the customer portal, e-mail signatures, case IDs, attachment information, and various other forms of noise that were automatically generated or inputted by support personnel. HTML tags were also removed using the Python package Beautiful Soup. Apart from this, data not inherently dirty but not relevant for the sake of modeling and finding similarities were removed, such as links, dates, and times. Lastly, any duplicated information in a ticket caused by human error was removed and whitespaces were fixed.

3.2.3 Evaluation & Selection of Machine Learning Models

A literature study was conducted for evaluating and selecting the most suitable state of the art NLP-model to use for the task. As the study focused on textual data and semantic text similarity, suitable models in regard to this were in focus. When comparing models, performance on this type of task was the primary consideration. Due to constraints in time and computational costs for the implementation of multiple models, it was not feasible to implement a wide variety of models and compare them empirically.

During the literature review in section 2.2.3 it was identified that for the task of semantic text similarity, deep neural network models outperformed any other type of model. In particular, autoregressive deep neural networks with a transformer architecture achieved state-of-the-art performance. Various models achieving state-of-the-art performance that fall into this type of model were identified during the literature review, such as GPT-3 developed by OpenAI, XLNet, and various models built on the BERT architecture developed by Google. Because of this, these types of models were primarily considered, and in particular XLNET, BERT, and GPT. These three models were considered as variants of these models have achieved state-of-the-art performance in various NLP tasks. In addition to this, the models are well-documented and easily accessible. As this area and these models are relatively new, there is currently a lack of peer-reviewed papers on many of the models, and in particular there is a lack of recent comparative studies on e.g. the previously mentioned models. Thus, the choice of the specific model (belonging to the family of autoregressive deep neural networks with a transformer architecture) to be used in this implementation does not have sufficient academic support to be the best model. However, as discovered in the literature review, GPT-3 has achieved state-of-the-art performance at various tasks, beating other transform-based models such as various BERT variants.

Based upon this, and due to the ease of accessibility and detailed documentation, GPT-3 was selected as the model to use in this project. In addition to this, GPT-3 was also the most recently released out of these models with the most number of parameters, indicating that it was the most powerful. GPT-3 had also shown good results without the need for extensive fine-tuning, even beating the previous fine-tuned state-of-the-art models without any fine-tuning at all, which was also an indication of great performance. Avoiding extensive fine-tuning was also very beneficial as huge models like GPT-3 are very computationally demanding to fine-tune.
3.2.4 Final Feature Selection and Hyperparameter tuning

Once a suitable model had been selected, various approaches were tested in terms of using different subsets of the available features to see if a simpler model can be constructed with better or equal performance than using all the kept features. To do so, a subset of 30 representative tickets was selected, for which 5 recommendations each were generated as this was the same number of recommendations to generate in the prototype. The decision to generate 5 recommendations is explained in section 4.1.7. The recommendations were generated in the same way the recommendations were generated in the final prototype.

This was evaluated manually by the researcher as the data was not categorical or labeled in any way, and thus there was no other way to systematically determine whether a new case and its recommended cases were related or not. Each recommendation was given a score of 1 if the recommendation was deemed relevant by the researcher, and 0 otherwise. As the subset was representative of the entire dataset, whether a recommendation was relevant or not could in most cases be determined without requiring any deep domain knowledge. The approach which received the highest score and thus generated the best recommendations was deemed optimal and selected. Additionally, the hyperparameters of the selected model were tweaked and determined using the same approach as above.

3.2.5 Deployment & Integration of Prototype

Lastly, the entire pipeline for generating recommendations was automated and a prototype to use for the evaluation was deployed. The pipeline was set up in Microsoft's cloud computing platform, Microsoft Azure. The pipeline was set up to perform each of the above steps sequentially. As a first step in order to be able to generate any recommendations, all available solved tickets were preprocessed, cleaned, embedded, and stored in Azure. The pipeline was set up to monitor any new tickets created and any updates to already existing tickets. When a ticket was created or modified, it was read from Dynamics into Azure where the ticket was preprocessed and cleaned, summarized, and embedded. If the ticket was new, or an update to a still open ticket, recommendations were computed by computing similarity using the cosine similarity between the ticket and the embeddings of previously closed tickets, and the generated recommendations were stored in Azure. On the other hand, if a ticket was updated and closed, instead of generating recommendations the ticket was instead preprocessed, cleaned, summarized, embedded, and lastly appended to the other embeddings of solved cases.

Once the generation of embeddings and recommendations was automated and running, a prototype web application was built for showcasing and evaluating the recommendations. The use of the prototype was integrated into the daily workflow of the support employees in order for them to be able to use it in their daily work and thus evaluate it. The web application displayed all open support cases with a hyperlink to the ticket in Dynamics, as well as the five recommended tickets with a hyperlink to their ticket in Dynamics. Additionally, the GPT-3 summary of each recommendation was also displayed alongside it as well as the cosine similarity score.

Lastly, the web application was deployed as an internal tool for the support personnel. The prototype was integrated in such a way that it would be feasible to use in their workflow, and a search function was implemented so that a support engineer easily could find the recommended cases for a particular case they are working on using the case ID.

All the steps in the pipeline were written in Python 3.10, and around 10 different packages were used. The most utilized and most important package used was Pandas, which was primarily utilized to manipulate data. The data that was ready from Dynamics was directly read into a Pandas DataFrame, which was the data object manipulated until the recommendations were generated and saved. In total, around 800 lines of code were written for the entire pipeline. The web application prototype was built using C# and Blazor.
3.3 Evaluation

The final part of the project was the evaluation of the decision support created by the implemented recommendation system. The impact the recommendations have on the support engineers’ daily work was evaluated in terms of how it affects the resolution time and efficiency. The evaluation was conducted qualitatively through user tests followed by interviews. A SUS evaluation was also conducted to obtain a quantitative metric on the user-perceived usability and satisfaction of the proposed new system. The methodology for the evaluation is presented below.

3.3.1 End User Evaluation

In qualitative research, a point of saturation is often reached and refers to the point where incoming data points such as interviews provide no new or very little insight and data relevant to the research question [26] [21]. Specifically for qualitative data collection from interviews, 6 interviewees can be enough to discover 80% of all qualitative themes and opinions on the subject [26]. Thus, it can be argued that more than 6 interviewees will lead to saturation as the amounts of new insights or data discovered by adding more users are heavily diminishing [25]. Based upon this, six employees from different user groups were selected to participate in the evaluation. Only employees who have previously contributed to the study during interviews and observations were selected, to be able to perform additional member checking to strengthen the validity of the study.

As was discovered during the pre-study in the interviews and observations as well as stated by the product owner, the most representative end users for the support division and for this study are the Second Line Application Specialists. Because of this, it was decided to have the same three Second Line Application Specialists from the prestudy participate during the evaluation. In total, there are 6 full-time second line support engineers at the Linköping office (and a lot of "part-time" second line employees if also counting engineers that are part of a product development team and only work with product support when necessary). Thus, if considering only full-time second line support engineers as they are the primary end users, the three users participating in the evaluation correspond to 50% of the second line support engineers at Sectra Linköping, which is an adequate sample to be considered representative of the entire population.

In addition to this, in order to include some diversity and gain other perspectives for the evaluation, users from three other representative user groups also affected by the change but to a lesser extent were selected. Thus, one test user was selected to participate in the evaluation from the first line support, as well as the Product Owner of the Dynamics service desk, and the Head of Scandinavian Support. All these test users were users that had participated during the prestudy. This selection of test users resulted in the most diverse and representative group of stakeholders suitable for the evaluation.

In order for the prototype and any effects it has on efficiency to be evaluated, the performance of the recommendations and the decision support provided was evaluated with the current system as the baseline. To generate data for the evaluation, support engineers and other relevant stakeholders were given access to the prototype, and be instructed to use it in their daily work, which was the recommended way to evaluate a system or prototype when it is feasible to test it in production as discussed in section 2.4. In order to generate adequate amounts of data, and to allow the subjects to gain a sufficient degree of experience with the new changes in order to provide insights to draw conclusions from, the system was used by the test users for five weeks. By deploying the system to be accessible in its intended environment, it can be evaluated in real-time on new data as this data is unseen by both the system and the users testing it, thus reducing the chances of introducing bias.
3.3. Evaluation

In addition to this, case studies where the end user can use a system or prototype in its intended way and target environment can be of great use when it comes to evaluating software systems and in particular decision support as the effects can quickly be studied, and suitable methods for qualitative collection and analysis were shown to be empirical methodologies such as interviews. Another reason why qualitative evaluations often are conducted for case studies and why this approach was selected is due to the fact that qualitative data generally is richer than quantitative data, as the amount of information that can be retained is higher for qualitative data. It can thus provide an in-depth exploration of a phenomenon in its natural contextual environment and enable better understanding and insights to be drawn.

As was established in section 2.4, there is significant support for evaluating both decision support and recommendation systems using a user-centric approach with qualitative methods, where the end users perceived opinion of the system is in focus. For this type of evaluation involving end users, qualitative methodologies such as interviews or questionnaires are recommended, and due to this, the evaluation was conducted using interviews with the end users after the evaluation period has ended.

In addition, to generate a quantitative metric on the perceived usability and satisfaction of the prototype, a SUS (System Usability Scale) score was also be elicited from the end users.

3.3.2 Qualitative Evaluation with Interviews

It was shown in section 2.4 that a commonly shared view when it comes to the evaluation of decision support was to not focus on the implementation or processes, but strictly evaluate it from the end users’ perception. The argument behind this was that as decision support is implemented solely to aid the decision-making of the user, this is also the perspective that should be in focus when evaluating it. A supported way to do this is to measure the perceived improved efficiency or increase in productivity from the end users’ perspective by self-rating. A similar argument can be made for the evaluation of recommender systems, where a user-centric evaluation using a qualitative approach was suggested. Based upon this, a user-centric qualitative approach was taken for the evaluation of this study.

With regard to the above points, the interviews focused on evaluating the end users’ perceived increase in efficiency. To gain a deeper understanding of what affects the change in efficiency, questions were also asked regarding the generated recommendations, as well as the decision support they provide as these are the underlying factors influencing the efficiency. The interviews were conducted after the five-week evaluation period, thus at the time of the interview all interviewees had been using the prototype integrated with their daily work for five to six weeks. All interviews were initiated by explaining the purpose of the study and the evaluation and how the collected insights will be used. In order to extract qualitative data and opinions from the users which could be aggregated and compared to base final conclusions upon, the interviews followed a strictly structured approach. All questions and discussion points were pre-defined, as well as their order. Both open and closed questions were prepared. This was done in order to have closed questions to collect absolute data regarding e.g. whether efficiency was increased or not, as well as more open questions inviting to a deeper discussion about the prototype. The flow of the interview roughly followed the pyramid model, where it began with closed questions and transitioned into questions that were more and more open towards the end of the interview. The full set of questions asked can be found in appendix E.

3.3.3 Quantitative Evaluation using System Usability Scale

To gather a quantitative metric on the user-perceived usability and satisfaction of the prototype, a SUS (System Usability Scale) evaluation was conducted. A SUS evaluation is by many seen as a cornerstone in the evaluation of any system or prototype when doing a user-centric evaluation, as discussed in subsection 2.4.4.
To complement the evaluation of how the prototyped recommendations may contribute to the efficiency of support personnel, as the change will alter their workflow and the current system, it is important to also evaluate this from a usability standpoint. Although, even though the implemented prototype could lead to improved efficiency, it would be counterproductive if this came at the cost of making the system less satisfactory to work in when it comes to usability or difficulty. Thus, to make sure no unwanted tradeoff between efficiency and usability would be introduced by implementing the prototype, the SUS evaluation was conducted to shed light on this. The data collection for the SUS evaluation was done after the five-week evaluation period, and every test user of the prototype was prompted to fill out a form containing the 10 standard statements. The 10 standard statements can be viewed in appendix H.
4 Results

The following chapter aims to present the findings from the study and its three phases. The chapter follows the same structure as the method chapter, and presents the results from the Pre-study and feature elicitation, Implementation, and lastly the Evaluation phase.

4.1 Prestudy and Requirements Elicitation

Below, the results obtained from the prestudy and requirements elicitation phase are presented.

4.1.1 Findings from the interview with the Dynamics Service Owner

The interview provided a great understanding of the support division, their workflow, and the system in use, as well as where to find more information about these topics. Important stakeholders to speak with were also identified, which were primarily the first and second line support as they are the end users working closely with Dynamics. This also meant that they were the end users of the prototype to be implemented, and could provide insights on what the development focus should lie on. Their workflow consists of the first line support engineers handling new cases, and the second line support engineers dealing with escalated tickets the first line could not handle. Typically, a large part of the support personnel’s workflow consists of searching for the solution to previously solved similar tickets with the hopes of finding a solution to the problem. Additionally, the current major issues with Dynamics were discussed, which mainly pertained to the poor search function in Dynamics, in particular as this is something heavily impacting the supports work. Lastly, it was verified that it is technically feasible to deploy a prototype and have it evaluated in production by allowing support personnel to use it for their daily work for some time. Based upon the above points, no major obstacles to the study were identified and thus could continue to the elicitation of requirements. The full transcribed responses to each interview question are presented in appendix B.
4.1.2 Findings from First Line Support observations

Below the findings of interest from the observation with the four first line support engineers are presented.

New support cases are either submitted directly by the customer in the customer portal, or by contacting the first line support engineers via phone or e-mail. If the support is contacted by e-mail, the support typically copy-pastes the provided information into the description field in Dynamics, whereas if the support is contacted via phone the information provided by the customer is manually entered. In addition, if the support is to manually register a ticket some additional information which is required by the system is provided, such as title, type of case, customer, system, product or product area, and responsible team. Additionally, if the ticket is submitted in Swedish (which is often the case for the Scandinavian support dealing with Swedish customers), the ticket description is also translated into English as this is the business language. There is no standard process for this, and some employees use machine translation whereas others write a translated summary by hand.

If a ticket is already opened and awaiting a response from the customer, the support employee assigned to the ticket is notified via e-mail once the customer has submitted a response. The e-mail notification contains the case ID, and the support engineer then typically copies the case ID into an internally developed tool called Sectra Redirect to immediately redirect to the corresponding ticket in Dynamics instead of having to search for it manually. The tool can also be used to search old case IDs from the previous support system, the HelpDesk, thus enabling both the new and old databases to be searched efficiently with just one search. The chain of replies from a customer after the initial case registration is saved as portal comments under a tab called action log in Dynamics. The support can also add internal comments in the action log with information that is not meant for the customer to view (e.g. sensitive data, or internal comments about the solution to the problem).

Each support employee has their own dashboard in Dynamics where they can keep track of their currently assigned cases, as well as various shared dashboards such as a dashboard for all open tickets, a dashboard for all unassigned tickets, etc. There are also dashboards for various types of tickets, such as a dashboard for cases awaiting a reply from the customer which should have a reminder sent to them. This is done manually, where a reminder is sent to the customer for a specific ticket via Dynamics after some predefined time has passed while awaiting a customer reply.

The first line support solves some types of cases themselves using documented instructions for common cases, typical cases are where a customer lacks authorization to e.g. delete images from a registered patient exam. Another typical case is where the customer site monitoring dashboard indicates that a system is having issues, this can often be manually solved by the first line support engineers proactively before the customer notices the problem and opens a ticket, such as the need to restart a service if there is a processing queue building up due to some error on the client’s side.

If the first line support can not solve a case, the case is escalated to the appropriate second line support team. Compared to the first line support, the second line support teams are more specialized in a particular product or product area, allowing for deeper troubleshooting. Typically, each major product or product area has its own second line support team that handles tickets that are escalated to them, in addition to developing the product. Apart from the product-specific second line support, there also exist more general second line support teams such as the second line application specialists.
4.1.3 Findings from First Line Support interviews

Below the findings of interest from the interviews with the four first line support engineers are presented. For the full responses and their frequencies, see appendix C.

For the first interview question *Is there any functionality or aspect of Dynamics you think works poorly, takes an unnecessarily long time to perform, or leads to mistakes in your work?* three topics were brought up by two or more employees and seem to be the most significant issues. These issues were regarding *slowness in Dynamics*, having to *resubmit attachments manually*, and the *poor search engine in Dynamics*.

Firstly, Dynamics is considered to be slow overall, with some actions taking a very long time. An example of this is accessing the audit log (the history of updates to a certain case) which typically takes around 10 seconds to load, sometimes longer. The main issue with the slowness of Dynamics comes from the impact on trivial tasks that are a part of the normal workflow, such as changing between different tabs in a dashboard or switching to a new support case, which is perceived as something strongly slowing down the work in Dynamics.

Secondly, attachments such as images which are sent to the support via email can not efficiently be uploaded to the corresponding support case due to limitations in Dynamics. When a customer submits attachments, the support engineer has to download all attachments to their local machine, and then upload them one by one to the corresponding support case in Dynamics.

Lastly, The search function in Dynamic used to search for previous support tickets does not work well, in particular when using keyword search, and thus makes it hard for support engineers to find previous tickets which may contain useful information.

Regarding the second interview question *If you could decide on new functionality to be added in Dynamics, what would you want?* two different topics were discussed, both of which were desired by two or more employees. These two points were in line with the responses from the first question, and the most demanded functionality was regarding *not having to resubmit attachments manually*, and something to *improve the poor search functionality in Dynamics*.

4.1.4 Findings from Second Line Support observations

Below the findings of interest from the observation with the three second line support engineers are presented. As some workflow processes are the same as the ones documented from the first line support observation, these have been left out from to reduce redundant information.

The use of Dynamics by the second line support is similar to the use by the first line support and their workflow is similar in many aspects. An identified major difference is that as the second line support does not register any new tickets themselves and only work with escalated tickets, some of the issues experienced by the first line support are not appearing in the work of the second line support, such as having to deal with attachments manually or having to manually refresh dashboards to not miss new tickets. Additionally, as the second line support work on more complex issues that often require more advanced troubleshooting, they spend less time in Dynamics and more time troubleshooting the problem and looking for a solution. Thus, issues with the slowness of Dynamics were not as prevalent for the second line support.

It was discovered that a shared issue with the first line support, is that of the poorly working search function in Dynamics. As the second line support engineers perform more complex troubleshooting, they rely heavily on finding previous solutions to similar problems by searching for cases in Dynamics. Thus, a lot of time needs to be spent searching in the
4.1. Prestudy and Requirements Elicitation

One major issue in the search process is that the initial support ticket and the sequence of portal comments (i.e. the conversation between customer and support as well as internal comments) are stored as separate objects due to the way Dynamics is designed. As there is no connection to go from the ticket to the portal comments or vice versa when searching in Dynamics, this can cause a lot of frustration and wasted time in some scenarios. As an example, if a support engineer would like to find the portal comments to a previously solved recognizable problem from a specific customer, there is no way to filter the portal comments search based on e.g. customer, which results in more search hits to go through. Another search-related issue, albeit not directly related to the functionality of Dynamics, is when a ticket is not properly translated or contains misspellings which may result in a keyword search not yielding a match.

Similarly to how the first line support escalated tickets they could not solve, the second line support can also escalate tickets to the third line support. When a case is escalated to the third line support, the running version of the customer’s software has to be manually entered by the support engineer. As different customers have their systems running on different versions, this must always be manually checked in Dynamics. However, when a customer has multiple different versions of the same software (e.g. different versions for production and testing) the support has to spend a lot of time finding the correct running version. As this is important information for the troubleshooting at the third line support, it would be beneficial to have this information automatically included, which would both prevent issues due to manually inputting the wrong version, as well as prevent time from being wasted searching for the customer’s version.

4.1.5 Findings from Second Line Support interviews

Below the findings of interest from the interviews with the three second line support engineers are presented. For the full responses and their frequencies, see appendix D.

For the first interview question Is there any functionality or aspect of Dynamics you think works poorly, takes an unnecessarily long time to perform, or leads to mistakes in your work? the second line engineers were very unified. Only the topic of the bad search functionality in Dynamics was mentioned by two or more employees. Similarly to what was discussed in the interviews with the first line support engineers, all three second line support engineers were not content with the search engine in Dynamics. In particular when it comes to poor performance when using keyword search, but also due to the lacking connection between the support case and the chain of portal comments as was discovered during the observations.

Regarding the second interview question If you could decide on new functionality to be added in Dynamics, what would you want? the second line was also unified. Functionality to help compensate for the poor search functionality in Dynamics was the most wanted functionality, which was desired by all three employees. This was the most desired feature, as this is where a lot of unnecessary time has to be spent and thus where the largest gains in terms of efficiency and time savings could occur. The primarily discussed idea for this is to based on the textual contents of a ticket identify the issue and find previously solved cases containing a potential solution to the problem. A predefined number of previous cases could then be suggested to the support engineer in Dynamics, with clickable links redirecting to the previous case. If successful, this would significantly reduce the amount of time needed to be spent searching for a solution to a problem which has previously already been solved.
4.1.6 Findings from the interview with the Head of Scandinavian Support

During the observations and interviews, the most frequently mentioned issue was regarding the search process in Dynamics, and something to deal with this was the most desired feature, such as a recommendation system. Throughout the data collection, as more qualitative data was generated and the aggregated findings continuously analyzed, this also seemed to be the aspect with the largest room for improvement. During the interview with the head of support, it was verified that out of the various possible approaches, the implementation of decision support through a ticket recommendation system would lead to the largest impact. It was also confirmed that keyword search and searching for previously solved cases is one of the biggest and most time-consuming issues for the support. Thus, the selected feature to implement was deemed both relevant and feasible. It was also confirmed that there are no blockers for the project in terms of technical aspects, or regarding the evaluation together with selected support engineers.

4.1.7 Member checking to decide on ticket recommender feature

The biggest identified issue among the first and second line support engineers, as well as the product owner of the support’s service desk and head of support, was the problematic search function which was identified to have a large impact on the support personnel’s daily work as this slows them down in the process of finding similar cases and their solutions. As this showed to be the overwhelmingly largest issue among both the first and second line support, a new feature that recommends previously solved similar cases given a new case was proposed to be implemented as suggested by the second line engineers.

During the member checking with the first and second line support engineers, the presented findings as well as the proposed feature to implement were approved by both user groups. No new suggestions were posed, and none of the members thought any other feature should have higher prioritization. No suggested changes to the proposition were made. There was a discussion regarding how many recommendations to show per case in the prototype, where both user groups deemed five recommendations per case to be an appropriate amount, given that in the majority of cases this is approximately the amount of tickets that have to be searched through to find a solution if the recommendations were relevant.

Thus, the agreed-upon finding to implement was a ticket recommendation system that for each new Dynamics support case generates recommendations of previously solved support tickets.

4.2 Implementation of a Prototype Recommendation System

Below, the implementation of the recommendation system prototype is presented. The section will cover primarily the cleaning and feature selection, followed by model selection and tweaking, and lastly the deployment and integration of the prototype.

4.2.1 Training Data Acquisition and Initial Feature Selection

Initially, the data queried from the Dynamics 365 database consisted of roughly 104,000 unique cases, each containing 374 features. The support tickets had a tabular data form and the 374 columns contained numerical, textual, categorical, and boolean data. An initial feature selection was performed as a first preprocessing step where each column and its values were investigated. Any features containing redundant information or information which was apparent to be of no use for the modeling were removed. Discussions were also held together with the second line support engineers in order to get a better understanding of what type of information certain data fields contained, as well as which features could be of importance.
4.2. Implementation of a Prototype Recommendation System

It was discovered that out of the 374 columns, only 13 features contained textual data, and since we are interested in the similarity based on textual similarity between two texts all other features were discarded. The contents of these remaining 13 features containing textual information which may be important to the modeling were thus analyzed deeper. It became apparent that out of these features, two of them contained redundant information which could be extracted from other features, and three of the features were discarded as they were not relevant to the task (e.g. feature suggestions in a product) and consequently available in less than 1% of all tickets.

Ultimately, for the preprocessing and cleaning of the training data, roughly 104,000 data points were left, and the feature space was reduced to 8 features after the manual analysis. The selected features were the following: ‘Title’, ‘Description’, ‘Portal Comments’, ‘Internal Comments’, ‘Problem’, ‘Question’, ‘Cause’, and ‘Solution’. The Title and Description fields were present in 100% of all tickets, Portal Comments in 95%, Internal Comments in 90%, and Problem, Question, Cause and Solution fields were present in 80-90% of all tickets.

The following features were kept for the textual similarity modeling:

- **Title** - The title of a support case. A short and concise description of the problem initially submitted upon creating a ticket.
- **Description** - The description of a support case initially submitted upon creating a ticket. Typically a short problem description, but sometimes long and containing error messages or logs.
- **Portal Comment** - The chain of any additional messages between the support engineer and the customer after the ticket was created. Frequently very long and containing a lot of noise.
- **Internal Comments** - Internal comments made by support engineers for documentation, such as solution process. Frequently quite long and containing a lot of noise.
- **Problem** - The problem the ticket was about. Entered by the case responsible support engineer upon closing the solved ticket. Typically very short and concise.
- **Question** - The question the ticket was about. Entered by the case responsible support engineer upon closing the solved ticket. Typically very short and concise.
- **Cause** - The issue that caused the underlying problem of the ticket. Entered by the case responsible support engineer upon closing the solved ticket. Typically very short and concise.
- **Solution** - The solution to the problem. Entered by the case responsible support engineer upon closing the solved ticket. Typically very short and concise.

4.2.2 Data Preprocessing and Cleaning

As the thesis project was conducted together with Sectra’s Scandinavian support, support tickets from the Scandinavian tickets were primarily considered. In addition to these, tickets from the US and UK support divisions were also kept as they were in English. Since multilingualistic models would result in poor interpretability during the development and evaluation of the model, and poor machine translations would introduce too much noise (particularly due to potential domain-specific language), any remaining tickets were ignored. The tickets from these three divisions amounted to roughly 68,000 tickets, which was deemed to be a sufficient amount of data points to use for modeling. Out of these tickets, roughly 65,000 support tickets were resolved and nearly 3000 are still open. Due to the data being dirty and unstructured textual data, it was difficult to achieve perfectly clean data. Thus the data was
4.2. Implementation of a Prototype Recommendation System

cleaned until it was ready to use for the modeling of the recommendations algorithm, which was to a point where any sample of the training data could be easily read by a human and free of major noise.

Some noise proved difficult to detect and remove due to their low frequency and uniqueness, in particular various types of log messages, warnings, errors, or other long texts not generated by a human, which could have a magnitude of different formats. As NLP models are designed to understand and act upon natural human language and not long error messages or log files, these had to be removed in order to not introduce a lot of noise. As 68,000 tickets were available, this could not be done manually but had to be done systematically. Upon taking repeated samples and looking at the lengths of ticket descriptions, portal comments, and internal comments, it was shown that nearly all tickets in the 95th percentile contained copy-pasted logs or error messages. This corresponded to text lengths of above 4000 characters. Thus, if any of these fields had a text length of above 4000 characters it was removed. Thus, removing extreme outliers in terms of text length was the final step to removing any undetected noise in a ticket. Out of the 68,000 tickets, roughly 3000 tickets had a field with a text length above 4000 characters which consequently was removed.

After the cleaning, the average length of a ticket had been reduced from 2690 to 600 characters, and the median length from 2890 to 900 characters, and the tickets had gone from barely interpretable to relatively clean and understandable.

4.2.3 GPT-3 Model

As per the literature review, GPT-3 was selected as the most suitable model to use in this study. OpenAI offers a specific GPT-3 model for generating word embeddings, text-embedding-ada-002, which was utilized. Due to the selected model, additional preprocessing was required as the text-embedding-ada-002 model had an input limit of 3000 words. Thus, as an additional preprocessing step directly before generating the embeddings, GPT-3 was also used to summarize the support tickets. This also had the benefit of removing any undetected noise that may still be present in the tickets after the cleaning, as well as making the language of the cleaned ticket more human-like. This yielded even cleaner tickets, free from any noise in the majority of the cases. Additionally, this resulted in all tickets having the same length before the embeddings which made the tickets more comparable.

Lastly, as the metric to use for comparing the similarity between embeddings, cosine similarity was selected, as this similarity measurement had the largest support and proved to be the most feasible to use for semantic text similarity.

4.2.4 Final Feature Selection and Hyperparameter Tuning

The full model utilizing all available fields proved to be the best and resulted in the most relevant recommendations generated on the subset of 30 representative tickets. As fields containing irrelevant or redundant information had been removed in the cleaning step, only unique and relevant information remained. Thus, by removing any of the fields the performance was negatively impacted as important information was left out and the recommendations started deteriorating.

For the selection of hyperparameters to use for the GPT-3 model, a temperature of 0 was selected, a maximum number of tokens of 200, and a frequency and presence penalty of 0. Performance was identical using the minimum recommended number of tokens of 200 (where each word is approximately 1-2 tokens) when generating summaries, compared to using up to 1000 tokens. This also became obvious upon studying the generated recommendations, as for most cases the summary generated was not over 200 tokens, and in the cases it was over 200 tokens, when slicing the maximum number of tokens down to 200 it was simply
summarized more concisely. Using a lower amount than 200 tokens sometimes resulted in important information being left out of the summary. Based upon this, 200 was the selected maximum number of tokens.

The temperature was set to 0 as this made the generated summarizations deterministic, i.e. the same input always produced the same output, which is what we want for our use case as we do not want to introduce any randomness. Similarly, the penalty for frequency and presence of tokens was set to 0, as these parameters are intended for generating content on new topics which we are not interested in our use case, we just want the most accurate summary.

### 4.2.5 Pipeline for Generating Recommendations

In figure 4.1, an overview of the pipeline is visualized and presented, beginning with a new Dynamics support ticket as input and ending with 5 generated recommendations displayed in the prototype as the output.

![Figure 4.1: Overview of the final support ticket pipeline](image)

### 4.2.6 Prototype Used for the Evaluation

In figure 4.2, a screenshot from the web application prototype is shown. This is the same prototype that was used during the evaluation of the recommendation system. As can be seen in the top left corner, the prototype features an input field for the case ID. Upon entering and filtering by a case ID, the 5 recommendations for this case are displayed featuring the Dynamics link, title, customer, ticket summary, and the similarity score.
4.3 Evaluation

Below, the results obtained from evaluating the prototype are presented. First, key findings from the qualitative interviews are presented, followed by the results from the SUS evaluation.

4.3.1 Qualitative Evaluation with Interviews

The key results from the qualitative evaluation with interviews are presented in the subsection below. The full list of questions can be found in appendix E, and the individual responses of each test user are available in appendix F. The aggregated responses and their frequencies for each of the 17 interview questions are available in appendix G. Below, the key findings are aggregated and summarized. The results for some key questions are also visualized. The results are further discussed in subsection 5.1.1.

When it comes to the results from the qualitative evaluation of the interviews (see appendix F and appendix G), in most aspects the results showed that the prototype yielded positive benefits even in its current stage. In terms of the goal of the study of finding something which may increase the efficiency of the support, the evaluation shows that in this case study, the use of machine learning to generate recommendations to provide decision support can indeed increase the support personnel’s efficiency in terms of reducing the resolution time. This was mainly shown in Q1, Q3, and Q4 where all 6 test users said that using the recommendations has increased their efficiency working with support tickets, enabled them to spend less time searching for similar cases and thus find a solution faster, and also allowed them to find cases which they could not find manually. This was also shown in Q2 where all 6 test users responded that the recommendations helped them to easier/faster find relevant previous cases.

Similarly, when asked about improvements in efficiency (Q5), 4 out of 6 test users said that their daily work had become easier since they gained access to the recommendations,
whereas the remaining two test users said that for them it was not the case since they are in managerial positions and thus their main work duties do not consist of working with solving tickets in Dynamics. When discussing the impact more broadly for the entire support division (Q12), 5 out of 6 users said that the main benefit would be the benefit of finding previous relevant cases and thus a solution faster/easier. Lastly, when asked whether the test users would like the recommendations to be implemented into Dynamics (Q16), all 6 test users said that this was something they would like to be implemented.

When discussing other use cases or user groups which were outside the scope of this study (Q13), but may still have use for the results from this project, the primary other use case mentioned was finding cases that could not be found manually. Additionally, test users believed that the recommendations could be of use to people outside of the support division such as customer managers or product owners who sometimes have to deal with customer issues or product issues and thus come in contact with support tickets. The test users believed that for their use cases, the recommendations could help provide insight and analytics. Additionally, it was mentioned that this could also be useful for onboarding new people to the support, as it could provide a good starting point for finding similar cases and the solution process for them.
4.3. Evaluation

Figure 4.3: Responses to question 6 - Has the quality of your work improved, remained the same, or declined since you started using the recommendations?

Figure 4.4: Responses to question 7 - From a scale of 1 to 5, how accurate do you believe the recommendations provided by the system are?

Figure 4.5: Responses to question 8 - Do you trust that the recommendation system is giving you relevant recommendations? Yes/No?
4.3. Evaluation

Figure 4.6: Responses to question 9 - When you perceive a recommendation as not relevant, do you believe this is because the system failed to find the most similar tickets, or because tickets of a similar nature simply do not exist?

Figure 4.7: Responses to question 2 - In what ways have the recommendations improved (or worsened) your efficiency?

Figure 4.8: Responses to question 14 - How does using the recommendations compare to other methods you’ve used to find similar tickets, e.g. keyword search?
4.3.2 Quantitative Evaluation using System Usability Scale

After conducting the interviews with the test person, they were prompted to fill in the SUS(System Usability Scale) form. For the full list of responses of each test user, please see appendix in table 4.9 the SUS-score for each test user is presented. The SUS score ranged from 75 to 90 for all six test users, which puts all individual SUS scores in the highest possible rating of A except for one which received a B rating, in accordance with the rating system presented in section 2.4.4. If we instead look at the mean SUS score of 86.25, it yields an A rating score and is well above the threshold of 78.9.

<table>
<thead>
<tr>
<th>Test User</th>
<th>SUS score</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Owner</td>
<td>90</td>
<td>A</td>
</tr>
<tr>
<td>Head of Support</td>
<td>87.5</td>
<td>A</td>
</tr>
<tr>
<td>SL Engineer 1</td>
<td>87.5</td>
<td>A</td>
</tr>
<tr>
<td>SL Engineer 2</td>
<td>90</td>
<td>A</td>
</tr>
<tr>
<td>SL Engineer 3</td>
<td>85</td>
<td>A</td>
</tr>
<tr>
<td>FL Engineer</td>
<td>75</td>
<td>B</td>
</tr>
<tr>
<td>Average</td>
<td>86.25</td>
<td>A</td>
</tr>
</tbody>
</table>

Figure 4.9: System Usability Scale score for each of the test users and the average SUS score.

If we instead look at each individual statement, the average SUS score converted from a 0-4 scale to a 1-5 scale ranges from 3.5 to 5 for each statement. See table 4.10 for the average SUS scores for all 10 questions.

<table>
<thead>
<tr>
<th>Question</th>
<th>Avg. SUS score (1-5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.83</td>
</tr>
<tr>
<td>2</td>
<td>4.67</td>
</tr>
<tr>
<td>3</td>
<td>4.67</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3.5</td>
</tr>
<tr>
<td>6</td>
<td>4.33</td>
</tr>
<tr>
<td>7</td>
<td>4.83</td>
</tr>
<tr>
<td>8</td>
<td>4.67</td>
</tr>
<tr>
<td>9</td>
<td>3.67</td>
</tr>
<tr>
<td>10</td>
<td>4.67</td>
</tr>
</tbody>
</table>

Figure 4.10: Average System Usability Scale score for each of the statements.

Out of these, only three questions had an average SUS score lower than or equal to 4.33, with the remaining average scores all being higher or equal to 4.67. The questions which received a low average score in relation to the other questions were the following:

- S5: “I found the various functions in this system were well integrated”
- S6: “I thought there was too much inconsistency in this system”
- S9: “I felt very confident using the system”

The results from the SUS evaluation are further discussed in subsection 5.1.2.
The following chapter aims to discuss the findings and results from the previous chapter. The chapter’s discussion will focus primarily on the obtained results, but the study’s method will also be discussed, as well as the work from a broader perspective and any ethical or societal implications it may have.

5.1 Results

Below, a discussion regarding the results obtained from the study is presented. First, the results obtained from the qualitative interviews will be discussed, followed by a discussion regarding the System Usability Scale evaluation.

5.1.1 Results from Interviews

When asked if the users would like to continue using the recommendations if they were integrated into Dynamics (Q11), 5 out of 6 users said that they would do it, whereas 1 user would do so only if the user did not immediately know what to do with a new ticket as this is the only scenario when it would be helpful. The reason behind this was that this user was a first line support engineer, and thus spent less amounts of time spent troubleshooting and searching among old tickets compared to other teams such as the second line support. Due to this, the recommendations could end up being distracting, in particular when not so relevant ones are recommended. Thus, the user would only consider looking at the recommendations when not knowing how to proceed with a current ticket.

When discussing other use cases or user groups which were outside the scope of this study (Q13), the primary other use case mentioned was finding cases that could not be found manually. This was due to the fact that the model generating the recommendations can understand synonyms and semantic similarities, overlook spelling mistakes, and understand multiple languages which regular keyword search does not. This had not been taken into consideration during the design of the study but was a great addition that also showed the benefits compared to the keyword search. Additionally, it was mentioned that this could also be useful for onboarding new people to the support division. The onboarding process is typically long as it takes a long time for users to learn the entire support ticket-solving process, as experience with Dynamics and some domain knowledge is needed. The recommendations
5.1. Results

could then be used to provide a good starting point for finding similar cases and learning the
solution process for them, thus speeding up the onboarding process.

The initial goal of the study was to increase efficiency without impacting the quality of the
work, which is why the approach to create decision support through the recommendations
was decided upon, rather than e.g. suggesting a solution directly to the customer which
would likely impact the quality negatively. In Q6, 3 out of 6 test users responded that the
quality of the work remained at the same level, whereas 3 out of 6 said that the quality had
actually improved. The fact that some test users perceived that the quality of their work had
actually increased was primarily connected to the fact that multiple users reported that with
the recommendations they could find relevant tickets they could not find manually or did
not know existed. Due to this, they could not only find a solution faster but sometimes also
a better solution, which consequently led to a higher quality of their work.

When looking at solely the relevance of the recommendations themselves (Q7), on a 1-5 scale
the average rating was 3.33. This is something that was discussed extensively during the inter-
views, and all users noted that due to various possible reasons, all five recommendations
were not always relevant for each new case. When asked whether this was due to the algo-
rithm performing poorly, or because there simply were no previous relevant cases (Q9), 5
out of 6 test users responded that they generally perceived that this was due to the fact that
the case was too unique and thus there did not exist any good recommendations to gener-
ate. One user however perceived that this was rather due to the algorithm generating poor
recommendations in some cases, as the user experienced that in many cases where the rec-
ommendations were bad, there did in reality exist similar cases which would have been more
relevant to recommend.

When instead asking how happy the users are with the benefits from the recommenda-
tions (Q10) the average rating was 3.50. As revealed during the discussions, the reason why
this is slightly higher than the results from Q9 is due to the fact that as long as 1 out of the
5 recommendations is relevant, this can be useful, and it did not occur so often that all five
recommendations were irrelevant. Additionally, multiple users argued that even in the case
that poor recommendations are presented, it is quick to see this and ignore them. Based
upon this it seems that the users trust the algorithm and the recommendations generated,
and in Q8 5 out of 6 test users said that they trust the recommendations generated, whereas
1 user was skeptical and would not blindly trust the recommendations. This user was the
same user who would not like to continue using the prototype unless absolutely necessary,
as the user believed the generated recommendations were often not relevant and thus seen
more as a distraction than something useful. Because of this user’s perception of the quality
of the recommendations, the user did simply not trust the recommendations enough to rely
on them.

When asked what the test users perceived to be the worst aspect of the recommendations or
what they would want to see changed (Q15), one user said that they would like the algorithm
to be improved, and another user mentioned the fact that some recommendations are irrele-
vant is an issue. This has already been discussed previously, and with the feedback received,
there are things that could have been done differently. A current improvement to battle this,
would be to add a similarity threshold a ticket would have to reach in order to allow it to be
a recommendation. As the cosine similarity is used to compute the similarity between the
tickets, this is possible in the current solution and would not require any extensive changes
to the modeling or the algorithm. As most cases had at least one relevant recommendation,
finding a suitable threshold to filter out all irrelevant recommendations and only display
the ones that are certainly good would result in a larger percentage of the recommenda-
tions being relevant. However, an issue with this is of course that by doing so relevant
recommendations could also be removed. Consideration would have to be taken in order to
find a balance, in particular as most test users mentioned that a poor recommendation did not impact their work in any significant way as they could just easily discard it, whereas accidentally filtering out a few great recommendations could have a heavier negative impact on the efficiency. Consequently, a change like this would improve the accuracy, but could potentially worsen the efficiency gains if a cosine similarity threshold can not be found that only discards irrelevant recommendations.

Lastly, when discussing additional feedback or suggestions for improvements (Q17) various interesting aspects were mentioned. Firstly, two users suggested that they believe being able to “like” or verify a recommendation could lead to great improvements of the model. Thus, it would be interesting to see the performance if a feedback loop was implemented where the recommendations are not purely based upon the textual similarity, but also an additional feature connected to the feedback from the support. One user also suggested including information from images in the model, as the user believed this could in some cases make the algorithm misinterpret the contents of a ticket as some tickets contain a lot of information within images and/or screenshots. Another user suggested to also include portal comments and internal comments when embedding old cases, and not only new cases, as although they to a large part contain redundant information, in the cases where they do not a lot of information could be missed.

All of the above aspects are things which could have been discovered and considered during the design and implementation of the recommendation system, but were not. However, neither of these things were observed during the observational sessions with the first and second line support. It was also not discovered during any of the interviews during the requirements elicitation phase. This and what could have been improved in the method to avoid this is further discussed in section 5.2.

5.1.2 Results from System Usability Scale (SUS) Evaluation

Out of the six test users, the average SUS score was 86.25 yielding an A rating. However, for the individual SUS scores of each test user only five out of six scores yielded an A rating, whereas one test resulted in a B rating with a score of 75. Out of the five A ratings, the SUS scores ranged from 85 to 90 with an average of 88.

When analyzing each individual response of the test user (test user 6, First Line Support Engineer) whose SUS score yielded a B score more thoroughly, there are no clear outliers among the score for each of the ten statements from this user when compared to the mean score for each statement. Thus, it seems that there are no specific statements that received a notable low score from this user, and each individual statement received a score of two or higher on the 0-4 scale. It seemed that many statements simply had a score slightly lower compared to the mean of the other five SUS scores which resulted in a lower SUS score for this user. This comes as no surprise, as this was the same user who although positive about the benefits, expressed some skepticism toward the recommendations and their relevance as discussed in the above section.

One potential reason for the notably lower score is one that became apparent during the interviews for the qualitative evaluations, where it was stated that the way the first line works with cases in Dynamics differs from other user groups as their work is more standardized and contains less manual troubleshooting. Due to this, as was discussed earlier, the recommendations will not be of help in the majority of the cases as the recommendations are simply not needed to progress in their work. Consequently, the benefits of the prototype did not yield the same level of increase in efficiency as the primary issue the prototype solves is not as prevalent within the first line support team.

As the added feature did not contribute to improving the user’s efficiency or improvement in daily work to the same extent as the other users, and thus it is likely that due to this
the additional feature was more seen as a distraction. Because of this, it is not strange that the perceived usability and satisfaction for the user were not as high as this is precisely what the SUS metric measures, as the possible efficiency gains in a minority of cases did not seem to outweigh the loss in usability or satisfaction that comes as a result of bloating the system with another feature that can cause distractions.

When it comes to the individual average scores for each SUS question, it was shown that the statements 5 - "I found the various functions in this system were well integrated" and 9 - "I felt very confident using the system" received a notably lower average score than the remaining question, at 3.5 and 3.67 respectively. In addition to this, statement 6 - "I thought there was too much inconsistency in this system" also received a slightly lower average score at 4.33 compared to the remaining statements, all of which had a score of 4.67 or higher. Although none of these scores are alarming and the overall average score is still very high, in particular statements 5 and 9 differentiate notably from the rest and may be worth looking further into.

The fact that statement 5 received the lowest average score at 3.5 is no surprise, as the current prototype is a separate web application from the real live system. This is also something mentioned by several test users in the interviews for the qualitative evaluation, as many of them stated that they would have preferred to have the recommendations integrated into Dynamics rather than as a stand-alone tool. Albeit it is easy to search for recommendations by case ID in the prototype and receive hyperlinks to the recommendations in Dynamics, the workflow is altered and impacted negatively as additional steps are required. By having the prototype integrated with Dynamics, the functionality would be more accessible and this point would likely improve.

Regarding statement 9 which had an average score of 3.67, nothing was explicitly stated in the interviews that would indicate what might cause this. Presumably, this statement is tightly connected to the same reason which caused the low score for statement 5, as these two statements have a similar average score that is notably lower than the rest. Having the prototype deployed in a new unfamiliar web application could be the reason why the test users did not feel very confident using it, as everything from the UI to the functionality was new compared to the current system. Thus, this point would likely also improve if the prototyped feature was integrated into Dynamics.

Regarding statement 6 with an average score of 4.33, the topic of inconsistency in the recommendations is something that was discussed during each interview with the test users. It was concluded earlier in this section and the previous section that overall the recommendations work well, but not in every case as sometimes the generated recommendations are irrelevant. Because of this, it is reasonable that the statement regarding consistency did not receive a very high score. As the root cause for this is known, this would be improved by further refining the algorithm responsible for generating the recommendations. Alternatively, as was discussed earlier, implementing a threshold for the similarity would reduce the number of irrelevant recommendations.

Overall, despite the above outliers in terms of grading and average scores, the overall average SUS score of 86.25 is a strong result well above the threshold for being classified as an A grade score. With a future implementation of the prototype directly into Dynamics and with some tweaks to which recommendations are displayed, the score would most likely improve even more. Even the current prototype results are showing indications of high user-perceived usability and satisfaction, and there is no indication that the proposed changes in the prototype will yield poor usability or satisfaction. However, the current solution and the prototype does have some issues which resulted in one of the scores yielding a B-rating. The most notable limitations are similar to what was discussed from the interview results, and the low score of statements 5 and 9 following due to the non-existing integration with Dynamics, as well
as statement 6 regarding the inconsistency in the recommendations are weak points in the currently prototyped solution.

5.2 Method

Below, a discussion regarding the methodology used in the study's implementation and evaluation is presented, as well as the aspects of replicability, reliability, and validity.

5.2.1 Implementation of the Recommendation System

It was discovered during the qualitative evaluation that some potentially important information was overlooked during the implementation of the algorithm. In particular, it was discovered that portal comments and internal comments which were only used for the embeddings of new cases could also provide valuable when embedding old cases as they do not always contain redundant information. Currently, it is assumed that they always carry redundant information for old cases, thus for some tickets important information is missing which will consequently affect the recommendations generated. Additionally, it was also discovered that some cases which contain images do not contain the same information in the text, which also led to potentially useful information being discarded.

In order to prevent this it could have been fruitful to during the implementation of the recommendation algorithm have an ongoing feedback loop from a domain expert regarding what fields and what information in the support tickets is important. By doing so, details like these which are easily overlooked could have been caught and analyzed further, potentially yielding better results in the few cases where any of this information would have an impact.

As none of these things were captured with the current elicitation techniques, observations and interviews, it is also possible that the method used was not adequate enough to capture all important aspects. Thus, more participants, longer observational sessions, or more in-depth interviews, could have led to discovering these aspects which would have affected the outcome of the study.

5.2.2 Evaluation of the Prototype

As the evaluation is primarily qualitative and limited to six test users, it is not possible to draw any statistically significant conclusions. Performing a deeper quantitative evaluation was hindered primarily by the fact that it was not feasible to implement the recommendations directly into the live production system, and thus an external tool had to be developed for the prototype. This made it not possible to benchmark different metrics such as the average resolve time between the current system and the new system if the feature were to be implemented. Additionally, given the limitations in time and resources in terms of test users, the amounts of data generated for five weeks would also not be enough as too much randomness can be introduced to a metric such as the average resolve time as this can vary drastically depending on the complexity of the case.

To improve the evaluation and find better support for the results, implementing the recommendations into Dynamics and having a larger number of people use it for a longer time in production would yield stronger results in terms of evaluating the improved efficiency, as enough data would be generated to draw statistically significant conclusions. Currently, with the qualitative evaluation we have shown that the efficiency was improved in terms of reduced resolve time for some users but not all, and it is not possible to put a number to the improvement.
5.2.3 Replicability, Reliability, and Validity

To enable replicability, the research method has been thoroughly described and documented for all parts of the study. During all forms of requirements elicitation and qualitative evaluation with test users, the process has been explained in detail and all questions used for interviews as well as the roles of the participants have been explicitly stated. Thus, replicating the method used in this study should grant similar results, even if using new test users or new data for the evaluation.

Due to the replicability, and since it is clear how the data was collected and used to arrive at the results [55], another researcher could at a later point repeat the study and obtain the same results given that it is conducted in the same context and domain. Thus, the reliability of the study is ensured given that the study is repeated in the same domain. If the study was instead conducted in another contextual domain, such as one not using support ticket data and Dynamics or a company in another industry, it can not be guaranteed that the elicited requirement and consequently the results obtained would be the same.

A lot of planning was put in place to ensure a high degree of construct validity and internal validity [55] where a representative context and representative participants with high domain knowledge were selected both for the requirements elicitation and the final evaluation [60]. As was discussed in section 2.4, the most representative set of test users available was selected to ensure a high degree of validity. Caution was also taken to avoid the Hawthorne effect during the observations, where the researcher alternated between actively participating and observing. Findings were also continuously verified with the users involved, i.e. by conducting member checkings, to further increase the construct validity.

As the study was conducted at a single company within a specific domain, it is difficult to say how generalizable the findings are as the obtained results may not hold true in a completely different domain, which reduces the external validity. However, as part of the research method is to elicit requirements in terms of what could increase the efficiency of the support personnel in the particular context which shapes the study, the results should be generalizable to another domain based upon this, thus strengthening the external validity. To strengthen the external validity and not only rely on qualitative data for a single case study, triangulation was also performed during the evaluation where in addition to the qualitative interviews a SUS survey was conducted.

5.3 Ethical and Societal Aspects

In a paper on guidelines for conducting case studies, Runeson and Höst argue that the ethical aspects of the study must always be considered [55]. This holds especially true where the case study deals with any confidential data which often is the case. Due to this, dealing with sensitive or confidential data must be done with careful consideration and approval of the company at which the case study is being conducted. To avoid spreading any confidential and sensitive information, this thesis has been thoroughly reviewed to make sure it contains no sensitive information.

Additionally, Runeson and Höst argue that any form of data collection between the researcher and participants such as interviews or observations must be explicitly agreed upon, as this is something important primarily for the long-term trust in case study research for the entire industry. In particular, this is important where a participant may want to be anonymous, or when the anonymization still makes a participant identifiable. To comply with this, all forms of interaction between the researcher and the participant which contained any form of data collection always explained the purpose of the study itself, as well as how and what the data will be used for. Additionally, what was going to be in the report was presented for approval before publishing the report.
The rise of applied AI and ML has introduced a lot of ethical discussions as they are integrated into more and more domains, in particular when it comes to the introduction of bias [1]. More and more legislation is being shaped to make unethical use of data and AI illegal, and in the US the Federal Trade Commission has stated that selling or using systems containing biased AI can be a violation of federal law. Thus, it is important to be able to show how a model built on top of AI and ML can arrive from its input to the produced outputs [1]. In order to do so, apart from trying to avoid introducing any bias during the modeling and implementation phase of the thesis, work has also been made in order to introduce full transparency to enable the monitoring of the model. For this work, this has been made sure by making the algorithms accessible and well documented, as well as by having a thorough demonstration of how the model reaches its recommendations given an input, followed by a handoff of the project.

The ethical aspects are amplified when dealing with automated AI and ML used in healthcare [24]. Issues such as "Who is to blame when something goes wrong?" typically arise, which are particularly important when the results could affect the health of another human. In order to battle this, the thesis has from the start been designed to simply serve as decision support for the support personnel in terms of which cases to search for a solution in, and not as an automated decision-making system with the purpose of solving an issue without human input. Thus, as all generated recommendation tickets are fed to a Sectra employee and no automatic action is taken based on this, there is no risk of endangering a patient.

There are also legal aspects to consider regarding AI, and the EU is working on new rules in their AI Act to ensure the ethical development of AI in Europe [47]. This is primarily to enforce transparency to be able to properly monitor and manage the risks that come with AI and its rapid developments.

From a societal perspective, AI and ML can help improve healthcare by providing cheaper, faster, or better healthcare [58]. In a use case like the one in this study, the goals of increased efficiency will also lead to the customers, which are hospitals, receiving help faster, which will lead to healthcare improvements.
6 Conclusion

The primary aim of this thesis was to examine how customer support efficiency could be improved through decision support powered by machine learning by enhancing a regular CRM support system. In order to achieve this aim, the following research question was postulated and researched: How can machine learning be used to create decision support in customer support, and how does this affect the efficiency of customer support? The question was answered by first examining what could lead to increased efficiency, followed by the implementation and evaluation of this.

From the prestudy, it was concluded that a recommendation system generating and recommending support engineers with previously solved support tickets was a desired feature to implement in a customer support context. During the implementation and evaluation of the work, there was some evidence that the use of state-of-the-art machine learning models for semantic textual similarity can outperform keyword searches. This was particularly shown in this context when working with unstructured dirty textual data where the understanding of semantics, different languages, and spelling errors became increasingly important, such as the support ticket data in this study.

In terms of the goal of the study to increase the efficiency of customer support, the evaluation from this case study shows that the use of machine learning to generate recommendations and provide decision support can most likely increase the support personnel’s efficiency in terms of reducing the resolution time of tickets. The above statement was supported by all six test users participating in the evaluation of this study, who all believed that the recommendations had increased their efficiency and enabled them to spend less time manually searching for similar cases and thus find a solution faster.

In addition to these benefits, additional use cases and user groups outside the scope of the study were discovered during the evaluation, such as finding cases that would not be possible to find manually, onboarding new employees, or enabling deeper analytics for product owners.

Lastly, based on the SUS score it was concluded that the user-perceived subject usability and satisfaction from using the prototype was higher than average for the majority of users, indicating no loss in either of these two aspects due to the proposed changes. Thus, it can also be concluded that implementing decision support through a recommendation system in
a customer support context does not cause any unwanted tradeoffs between usability and efficiency.

One should keep in mind, however, that both of the evaluations were only done based on 6 individual users.

To conclude, this study shows some support for the claim that machine learning can be used to create decision support in a customer support context with the goal of increased efficiency. This is shown by implementing decision support through a ticket recommender prototype powered by machine learning, resulting in a reduction in ticket resolution time for a majority of the users examined, and thus increased their efficiency.

6.1 Future Work

Although the criteria for success for the thesis was fulfilled, there still exist things that could have been performed differently or improved upon, particularly if any time and resource constraints were lifted. As the prototype received great feedback and proved potential to increase the efficiency already as a stand-alone tool, it would be interesting to see what the results would be and how much they would improve if the prototype was implemented directly in the Dynamics client the support personnel are using when working. A further step given a real implementation would also be to conduct a longer evaluation with more test users in order to gather quantitative data for further analysis which could support the evidence found in this study. If sufficient amounts of data points would be generated, quantitative metrics such as the average resolution time of a case could be evaluated before and after the change, and statistically significant conclusions could be drawn from this. For an even finer comparison, one could implement functionality that only monitors the time a case is being actively worked on by the support, so as to not factor in time waiting on replies from the customer or other aspects that may affect the resolve time but are out of the scope. Transparency of the evaluations would be essential so that even negative indicators, where the added functions would be detrimental to resolving the cases would need to be carefully collected.

It would also be interesting to investigate if even better results could be obtained by implementing another dimension than textual similarity, by selecting and/or engineering new features to be integrated into the model. E.g. by specifying weights for the different fields used for the textual similarity and for the new features. An interesting feature to add would be a feedback loop where end-users could "like" or "dislike" a recommendation or case, which would affect the prioritization of the recommendations generated.

Lastly, it would be interesting to see how well this same solution would work in other domains, and see how well, if at all, the shown benefits from this study could generalize to customer support in other fields.


[27] Aminul Islam and Diana Inkpen. “Semantic Text Similarity Using Corpus-Based Word Similarity and String Similarity”. In: ACM Transactions on Knowledge Discovery from Data 2.2 (July 2008). ISSN: 1556-4681. DOI: 10.1145/1376815.1376819


[34] Irene Li, Jessica Pan, Jeremy Goldwasser, Neha Verma, Wai Pan Wong, Muhammed Yavuz Nuzumlalı, Benjamin Rosand, Yixin Li, Matthew Zhang, David Chang, R. Andrew Taylor, Harlan M. Krumholz, and Dragomir Radev. “Neural Natural Language Processing for unstructured data in electronic health records: A review”. In: *Computer Science Review* 46 (2022), p. 100511. ISSN: 1574-0137. DOI: [https://doi.org/10.1016/j.cosrev.2022.100511](https://doi.org/10.1016/j.cosrev.2022.100511)


[38] Long Ma and Yanqing Zhang. “Using Word2Vec to process big text data”. In: *2015 IEEE International Conference on Big Data (Big Data)*. 2015, pp. 2895–2897. DOI: [10.1109/BigData.2015.7364114](https://doi.org/10.1109/BigData.2015.7364114).


[60] Christopher B. Seaman. “Qualitative methods in empirical studies of software engineering”. In: *IEEE Transactions on Software Engineering* 25.4 (1999), pp. 557–572. DOI: [10.1109/32.799955](https://doi.org/10.1109/32.799955).


A Interview Questions for the Service Owner Interview

List of interview questions

1. What does the support workflow look like?
2. Where do you think there is most potential to create value?
3. Which stakeholders are the most relevant to elicit requirements from?
4. Which part of the support personnel’s work in Dynamics do you think is taking the longest time or causing the most problems?
5. What type of feature do you think would be the most helpful in helping the support personnel perform their work?
6. How are support tickets created and how are they saved in Dynamics?
7. Are there any available APIs that can be used for querying the data from Dynamics?
8. For evaluating the results, would it be feasible to deploy a prototype and let support personnel use it for their work for some time?
Interview Responses from the Service Owner Interview

List of interview questions and their responses

1. What does the support workflow look like? - The support division consists of three levels of support engineers, the first, second, and third line. The first line support engineers are the first to handle a new case as soon as possible after a case is registered. The first step of the process is to make sure that sufficient information to be able to solve the issue has been submitted by the customer, and if not, contact them to complement missing information. Next, the first line support engineers solve the problem directly, or escalate it to the appropriate second line support team. Similarly, if the second line support engineer team can not solve the issue, the ticket is escalated to the appropriate third line support team.

2. Where do you think there is most potential to create value? - The PO believes that there is room for improvements primarily for saving time by reducing mundane tasks for current employees, but also improvements which can facilitate the onboarding of new users.

3. Which stakeholders are the most relevant to elicit requirements from? - Relevant stakeholders identified are those the support teams that work closely with Dynamics, which are the first line support engineers and second line support engineers, in particular the application specialists.

4. Which part of the support personnel’s work in Dynamics do you think is taking the longest time or causing the most problems? - An important part of the support personnel’s workflow consists of searching for the solution to previously solved similar tickets. Thus, the PO believes that the largest issue as of right now is the poor performance of the search function in Dynamics, resulting in a lot of time spent trying to locate previous solutions to similar problems. In addition to this, the previous support tickets are divided between two different databases, the current one and an older database which was used with the previous system, which further complicates the search for a solution.

5. What type of feature do you think would be the most helpful in helping the support personnel perform their work? - Something that would allow the support engineers to more easily search for or find a potential solution among previous tickets.
6. *How are support tickets created and how are they saved in Dynamics?* - Support tickets are either created directly by customers in the customer portal, or by first line support engineers when they receive an email or call from a customer. The tickets are stored in an Azure SQL Database.

7. *Are there any available APIs that can be used for querying the data from Dynamics?* - The PO states that the default OData v4 APIs can be used for accessing the Dynamics data.

8. *For evaluating the results, would it be feasible to deploy a prototype and let support personnel use it for their work for some time?* - The PO confirms that it would be possible to deploy a prototype and allow a couple of employees to use it in production for some time for evaluation purposes.
The answers to the question are further explained below based upon the transcribed interview responses.

- Dynamics is considered to be slow overall, with some actions taking a very long time. An example of this is accessing the audit log (the history of updates to a certain case) which typically takes around 10 seconds to load, sometimes longer. The main issue with the slowness of Dynamics comes from the impact on trivial tasks that are a part of the normal workflow, such as changing between different tabs in a dashboard or switching...
to a new support case, which is perceived as something strongly slowing down the work in Dynamics.

- Attachments such as images which are sent to the support via email can not efficiently be uploaded to the corresponding support case due to limitations in Dynamics. When a customer submits attachments, the support engineer has to download all attachments to their local machine, and then upload them one by one to the corresponding support case in Dynamics.

- The search function in Dynamic used to search for previous support tickets does not work well, in particular when using keyword search, and thus makes it hard for support engineers to find previous tickets which may contain useful information.

- The dashboards in Dynamics does not automatically refresh upon events. E.g. if a new support case has been registered, it will not appear in any of the dashboard lists until the support engineer manually refreshes the page.

- Reminders for open tickets where the support is waiting on further information or confirmation that the issue has been solved have to be manually sent by the support engineer. Thus, the support has to periodically check a dashboard which contains tickets that have been waiting on customer response for a predefined amount of days, and manually send a reminder on each of these tickets.

![Bar chart showing responses to question 3](image)

Figure C.2: Responses to question 3 - *If you could decide on new functionality to be added in Dynamics, what would you want?*
Interview Responses from Second Line Support

Figure D.1: Responses to question 1 - *Is there any functionality or aspect of Dynamics you think works poorly, takes an unnecessarily long time to perform, or leads to mistakes in your work?*

Below each answer is explained in greater depth.

- Similarly to what was discussed in the interviews with the first line support engineers, all three second line support engineers were not content with the search engine in Dynamics. In particular when it comes to poor performance when using keyword search, but also due to the lacking connection between the support case and the chain of portal comments.

- Attached images which often contain screenshots of error messages or logs are not searchable by text, causing relevant tickets to be missed in the search.

- The need for manual input of certain fields, such as the running version of the customers system, currently requires quite some time although this could feasibly be automated. Apart from being time-consuming, this also introduces the risk of errors due to relying on manual input.
The answers to the question are further explained below based upon the transcribed interview responses.

- The most desired functionality is something that would help to deal with the poor search engine, as this is where a lot of unnecessary time has to be spent and thus where the largest gains in terms of efficiency and time savings could occur. The primarily discussed idea for this is to based on the textual contents of a ticket identify the issue and find previously solved cases containing a potential solution to the problem. A predefined number of previous cases could then be suggested to the support engineer in Dynamics, with clickable links redirecting to the previous case. If successful, this would significantly reduce the amount of time needed to be spent searching for a solution to a problem which has previously already been solved.

- The automation for the input of certain fields such as the running version of the system, as described earlier, is also something highly desirable and would lead to time saved and fewer errors.

- More customizability in Dynamics when it comes to the layout, shortcuts, buttons et cetera, as Dynamic currently allows for some basic setup to be done, it is quite limited.

- Templates for generating responses for the most common cases, as there exist some common cases which have the same solution. Instead of having to repeatedly write the same instructions for a normal problem, a lot of time could be saved if a template for the identified problem could be selected instead. This could also, apart from saving time writing a reply once the solution has been found, reduce the time spent searching previous cases for a solution as it would be enough to select the appropriate template.

![Bar chart showing frequencies of answers to question 2:](image-url)

**Figure D.2: Answers and their frequencies to question 2 - If you could decide on new functionality to be added to Dynamics, what would you want?**
Interview Questions for the Evaluation

List of interview questions

1. Question 1 (Q1): Do you believe that your efficiency working with tickets has increased from using the recommendations? Yes/No?

2. Question 2 (Q2): In what ways have the recommendations improved (or worsened) your efficiency?

3. Question 3 (Q3): Do you believe that you need to spend less time searching for similar tickets and thus find the solution to a problem faster when using the recommendations? Yes/No?

4. Question 4 (Q4): Do you believe that the recommendations have helped you find relevant cases which you would otherwise not have found manually? Yes/No?

5. Question 5 (Q5): Do you believe that the difficulty of performing your daily work has decreased using the prototype? Yes/No?

6. Question 6 (Q6): Has the quality of your work improved, remained the same, or declined since you started using the recommendations?

7. Question 7 (Q7): From a scale of 1 to 5, how accurate do you believe the recommendations provided by the system are?

8. Question 8 (Q8): Do you trust that the recommendation system is giving you relevant recommendations? Yes/No?

9. Question 9 (Q9): When you perceive a recommendation as not relevant, do you believe this is because the system failed to find the most similar tickets, or because tickets of a similar nature simply do not exist?

10. Question 10 (Q10): How happy are you with the benefits the recommendations provide as they are today?

11. Question 11 (Q11): Would you want to continue using it in the future if it were integrated in Dynamics?
12. Question 12 (Q12): How do you believe the recommendations will impact the efficiency of the entire support division?

13. Question 13 (Q13): In what other contexts do you believe the recommendations would be most useful?

14. Question 14 (Q14): How does using the recommendations compare to other methods you’ve used to find similar tickets, e.g. keyword search?

15. Question 15 (Q15): Is there something you think is working poorly, or anything you wish would have been done differently or changed?

16. Question 16 (Q16): Would you like this prototyped feature to be implemented in Dynamics?

17. Question 17 (Q17): Do you have any feedback or suggestions for improving the recommendation system, or anything else to comment on?
Interview Responses from the Evaluation

Test User 1 - Product Owner of the Dynamics ServiceDesk

1. Q1 - Yes.
2. Q2 - Does not work with solving support tickets, but for the support working in Dynamics absolutely by easier finding old cases.
3. Q3 - Yes.
4. Q4 - Yes
5. Q5 - Does not work with solving support tickets, thus no.
6. Q6 - The same.
7. Q7 - 3.
8. Q8 - Yes.
9. Q9 - Difficult to say, does not know.
10. Q10 - 3.
11. Q11 - Yes.
12. Q12 - By allowing to faster and easier find similar cases, thus less time spent searching.
13. Q13 - Apart from previously mentioned, also the onboarding of new people. Allows them to easily find similar cases and study the whole process and flow of how the ticket can be solved. So the recommendations can be used for more than trying to find a solution, it can also be used for learning.
14. Q14 - Automated, faster, can recommend you tickets you didn’t exist.
15. Q15 - Deep knowledge is required in some cases to find a solution to a problem. If you are new and blindly trust the recommendations, you can get recommendations that seem feasible but in reality are not what you are looking for. This can cause issues.
16. Q16 - Yes, would like to see it integrated in a new tab, featuring the recommended cases, their titles, issues, causes, and solutions.

17. Q17 - Would also have liked to add portal and internal comments also for solved cases as they can contain a lot of hidden information.
Test User 2 - Head of Scandinavian Support

1. Q1 - Yes.

2. Q2 - Does not work with solving support tickets, but has an easier time finding similar cases for a known customer issue. For the support yes. Typically works really well, and in the cases where it doesn't you can quickly notice this and ignore the recommendations.

3. Q3 - Yes.

4. Q4 - Yes.

5. Q5 - Does not work with solving support tickets, thus no.

6. Q6 - The same.

7. Q7 - 3.

8. Q8 - Yes.

9. Q9 - Difficult to say, when I have seen strange recommendations I have not seen many of the original case. So I would guess because there exists no similar cases.


11. Q11 - Yes.

12. Q12 - By finding a solution faster, but also for other interested persons to gain insight.

13. Q13 - Apart from what is discussed previously, it can be used by basically anyone who is at some point looking at Dynamic cases. Primarily it could also be used by customer managers as well as product owners to gain insight on their products and potential issues.

14. Q14 - Much better, especially in the cases where you don’t know what to search on to start digging.

15. Q15 - 

16. Q16 - Yes, would like to see it integrated at the front page or in an individual tab.

17. Q17 - Be able to like recommendations, and thus affect the algorithm so that good cases are recommended more frequently.
Test User 3 - Second Line Support Engineer #1

1. Q1 - Yes
2. Q2 - Easier finding the solution to a case, or hints by finding information about a certain part of a product.
3. Q3 - Yes.
4. Q4 - Yes.
5. Q5 - Yes.
6. Q6 - Improved.
7. Q7 - 4.
8. Q8 - Yes.
9. Q9 - Because similar cases do not exist, or because the only info contained by the new case is too general.
11. Q11 - Yes I would. I will probably continue using it even if it is not integrated in Dynamics.
12. Q12 - Not only by finding solutions easier and faster, but also by finding cases you would otherwise not have found with a regular search which can save you a lot of time.
13. Q13 - Apart from previously mentioned things, I also think it would be useful for on-boarding new people.
14. Q14 - Like state above, can find cases you would otherwise not find.
15. Q15 - Sometimes the recommendations are not relevant, but as you can easily ignore them and continue as usual this is not something causing irritation. And the pros outweigh this.
16. Q16 - Yes, somewhere easily accessible.
17. Q17 - A way to include the information from images and screenshots would greatly improve the results in the cases where there is very little text but images.
Test User 4 - Second Line Support Engineer #2

1. Q1 - Yes.

2. Q2 - Would allow to easily get an overview if there seem to be similar cases with the same issue, and be able to see the potential solution or process for finding the solution can be used for the case I am working with.

3. Q3 - Yes.

4. Q4 - Yes.

5. Q5 - Yes.

6. Q6 - Improved.

7. Q7 - 4.

8. Q8 - Yes, but sometimes the same word is used for different problems, and in those cases the algorithm of course had a hard time differentiating between these.

9. Q9 - Because similar cases does not exist, but generally if no recommendations were not relevant this was the case, and thus you could also use this to quickly see if you are dealing with a new problem.

10. Q10 - 3

11. Q11 - Absolutely, it would in many cases save time, because it is difficult using the built in search engine to find relevant similar cases.

12. Q12 - By giving an alternative to the bad search function to easier and faster find cases to study.

13. Q13 - Apart form what is previously discussed, also be able to find cases and solutions to a problem which you did not know existed among the previous cases. But also by finding cases know exist, but somehow can not find manually.

14. Q14 - In most cases works better, but sometimes doesn’t but then you can quickly determine that the recommendations are not so relevant.

15. Q15 -

16. Q16 - Yes. I think it can be used and yield good results and apart from solving cases faster, also find case relationships. Also, in addition also display product and version number in Dynamics.

17. Q17 - I would like to verify/mark a case as related to another based upon the recommendation, so there will be a future way to find the link. Also, free text search would be nice to have, cause then managers etc could also use it to gain insight.
Test User 5 - Second Line Support Engineer #3

1. Q1 - Yes.
2. Q2 - Finding a relevant solution process faster.
3. Q3 - Yes.
4. Q4 - Yes.
5. Q5 - Yes.
6. Q6 - Improved, as more time freed up.
7. Q7 - 4.
8. Q8 - Yes.
9. Q9 - Probably because new type of problem.
10. Q10 - 4
11. Q11 - Yes!
12. Q12 - Would make it faster to arrive at a starting point for a solution.
13. Q13 - I think anyone to various extent who is every using Dynamics for looking at tickets.
14. Q14 - Both easier and faster to use.
15. Q15 -
16. Q16 - Yes
17. Q17 -
Test User 6 - First Line Support Engineer

1. Q1 - Yes.
2. Q2 - By recommending relevant cases to start looking into.
3. Q3 - Yes.
4. Q4 - Yes.
5. Q5 - No. As typically more standardized way of working not requiring to use the search functionality in Dynamics extensively.
6. Q6 - The same.
7. Q7 - 2
8. Q8 - No.
9. Q9 - I think it can be both.
10. Q10 - 3.
11. Q11 - Yes - If I would need to because I do not know how to deal with my current problem, then yes I would.
12. Q12 - For me not so much, but for other support teams more intro troubleshooting then it would make it easier for them to find the right cases.
13. Q13 - For support personell, but also for deployment engineers who are projec leading implementations at a site, as they have to deal with a lot of issues which typically are standard issues but to them are completely new and thus requires a lot of time.
14. Q14 - If good recommendations, saves time.
15. Q15 - Still needs some more work as at this state too few recommendations are accurate.
16. Q16 - Yes I would, I would like to see it improved but even in its current state it can be useful, as it doesn’t take a long time to disregard bad recommendations, but can be very helpful in the few cases it does help you.
17. Q17 - Would be interesting to see how many cases are matched as similar above some threshold, as then you could know how many prevalent a problem is.
Figure G.1: Responses to question [1] - Do you believe that your efficiency working with tickets has increased from using the recommendations? Yes/No?

Figure G.2: Responses to question [2] - In what ways have the recommendations improved (or worsened) your efficiency?
Figure G.3: Responses to question 3 - Do you believe that you need to spend less time searching for similar tickets and thus find the solution to a problem faster when using the recommendations? Yes/No?

Figure G.4: Responses to question 4 - Do you believe that the recommendations have helped you find relevant cases which you would otherwise not have found manually? Yes/No?

Figure G.5: Responses to question 5 - Do you believe that the difficulty of performing your daily work has decreased using the prototype? Yes/No?

Figure G.6: Responses to question 6 - Has the quality of your work improved, remained the same, or declined since you started using the recommendations?
Figure G.7: Responses to question 7 - From a scale of 1 to 5, how accurate do you believe the recommendations provided by the system are?

Figure G.8: Responses to question 8 - Do you trust that the recommendation system is giving you relevant recommendations? Yes/No?

Figure G.9: Responses to question 9 - When you perceive a recommendation as not relevant, do you believe this is because the system failed to find the most similar tickets, or because tickets of a similar nature simply do not exist?

Figure G.10: Responses to question 10 - How happy are you with the benefits the recommendations provide as they are today?
Figure G.11: Responses to question 11 - Would you want to continue using it in the future if it were integrated in Dynamics?

<table>
<thead>
<tr>
<th>Yes</th>
<th>Yes but only when other options are exhausted</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure G.12: Responses to question 12 - How do you believe the recommendations will impact the efficiency of the entire support division?

<table>
<thead>
<tr>
<th>Easier/faster to find previous relevant cases and a solution</th>
<th>Gain insight</th>
<th>Finding cases you would never find manually</th>
<th>Give an alternative to the poor search function</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure G.13: Responses to question 13 - In what other contexts do you believe the recommendations would be most useful?

<table>
<thead>
<tr>
<th>Finding cases not possible to find manually</th>
<th>Anyone in contact with Dynamics e.g. customer managers and product owners</th>
<th>Learning the ticket-solving process e.g. onboarding of new people</th>
<th>Deployment engineers dealing with simple issues during deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Can recommend tickets you didn’t know existed or when you don’t know what to search for.

**Figure G.14:** Responses to question 14 - How does using the recommendations compare to other methods you’ve used to find similar tickets, e.g. keyword search?

The algorithm for generating the recommendations needs to be improved.

Sometimes the recommendations are not relevant.

Blindly trusting the recommendations and pursuing the wrong path if the recommendation is incorrect.

**Figure G.15:** Responses to question 15 - Is there something you think is working poorly, or anything you wish would have been done differently or changed?

Yes - in a new tab.

Yes - somewhere easily accessible.

Yes - on the front page.

**Figure G.16:** Responses to question 16 - Would you like this prototyped feature to be implemented in Dynamics?
Figure G.17: Responses to question [17] - Do you have any feedback or suggestions for improving the recommendation system, or anything else to comment on?
System Usability Scale Statements

List of SUS statements (rated on a 5 Likert scale from "strongly agree" (1) to "strongly disagree" (5))

1. Statement 1 (S1): I think that I would like to use this system frequently.
2. Statement 2 (S2): I found the system unnecessarily complex.
3. Statement 3 (S3): I thought the system was easy to use.
4. Statement 4 (S4): I think that I would need the support of a technical person to be able to use this system.
5. Statement 5 (S5): I found the various functions in this system were well integrated.
6. Statement 6 (S6): I thought there was too much inconsistency in this system.
7. Statement 7 (S7): I would imagine that most people would learn to use this system very quickly.
8. Statement 8 (S8): I found the system very cumbersome to use.
9. Statement 9 (S9): I felt very confident using the system.
10. Statement 10 (S10): I needed to learn a lot of things before I could get going with this system.
System Usability Scale Responses from the Evaluation

List of SUS statements (rated on a 5 Likert scale from "strongly agree" (1) to "strongly disagree" (5))

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Test User 1 - Product Owner of the Dynamics service desk - 90

1. Q1 - 5
2. Q2 - 2
3. Q3 - 5
4. Q4 - 1
5. Q5 - 4
6. Q6 - 1
7. Q7 - 5
8. Q8 - 2
9. Q9 - 4
10. Q10 - 1

Test User 2 - Head of Scandinavian Support - 87.5
1. Q1 - 5
2. Q2 - 1
3. Q3 - 5
4. Q4 - 1
5. Q5 - 3
6. Q6 - 1
7. Q7 - 5
8. Q8 - 2
9. Q9 - 4
10. Q10 - 1

Test User 3 - Second Line Support Engineer #1 - 87.5
1. Q1 - 5
2. Q2 - 2
3. Q3 - 5
4. Q4 - 1
5. Q5 - 3
6. Q6 - 2
7. Q7 - 5
8. Q8 - 1
9. Q9 - 4
10. Q10 - 1

Test User 4 - Second Line Support Engineer 2 - 0
1. Q1 - 5
2. Q2 - 1
3. Q3 - 4
4. Q4 - 1
5. Q5 - 2
6. Q6 - 1
7. Q7 - 5
8. Q8 - 1
9. Q9 - 3
10. Q10 - 1

Test User 5 - Second Line Support Engineer 3 - 5
1. Q1 - 5
2. Q2 - 1
3. Q3 - 5
4. Q4 - 1
5. Q5 - 3
6. Q6 - 2
7. Q7 - 5
8. Q8 - 2
9. Q9 - 3
10. Q10 - 5

Test User 6 - First Line Support Engineer - 75
1. Q1 - 4
2. Q2 - 1
3. Q3 - 4
4. Q4 - 1
5. Q5 - 3
6. Q6 - 3
7. Q7 - 4
8. Q8 - 1
9. Q9 - 4
10. Q10 - 3