TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Large Language Model Based Chatbots in Industrial Maintenance Applications

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Master of Science (Dipl. Ing.)

unter der Leitung von

Univ.-Prof. Dr.-Ing. Fazel Ansari
(E330-01-02 Institut für Managementwissenschaften,
Forschungsgruppe Produktions- und Instandhaltungsmanagement)

Univ.-Lektor Dipl.-Ing. Linus Kohl
(E330-01-02 Institut für Managementwissenschaften,
Forschungsgruppe Produktions- und Instandhaltungsmanagement)

eingereicht an der TU Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Julian Kölbl, BSc

[redacted]

Wien, im Dezember 2023

[redacted]
Julian Kölbl

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Large Language Model Based Chatbots in Industrial Maintenance Applications

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Ich erkläre hiermit Eides statt, dass ich meine Diplomarbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen selbstständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, genannt habe.

Weiters erkläre ich, dass ich dieses Diplomarbeitsthema bisher weder im In- noch Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Wien, im Dezember 2023

[redacted]

Julian Kölbl

Danksagung

An dieser Stelle möchte ich all jenen Dank aussprechen, die mich durch mein Studium begleitet und während des Erstellens der Diplomarbeit unterstützt haben.

Ich möchte mich zuerst bei Prof. Fazel Ansari für die Betreuung und Begutachtung dieser Arbeit bedanken.

Des Weiteren gebührt mein Dank Herrn Dipl.-Ing. Linus Kohl, der mir das Verfassen dieser Arbeit erst ermöglicht hat. Durch seine fachliche und wissenschaftliche Unterstützung konnte ich mich in dieses spannende Thema einarbeiten und dieses vertiefen.

Bedanken möchte ich mich außerdem bei meinen Kommilitonen und allen mir nahestehenden Personen, die mich auch abseits des Studiums unterstützt und begleitet haben.

Abschließend möchte ich mich bei meiner Familie bedanken, die mir mein Studium ermöglicht hat. Besonderer Dank gebührt meinen Eltern, welche mir in allen Phasen, Höhen und Tiefen, mit Geduld beigestanden sind.

Kurzfassung

Als Reaktion auf den steigenden Fokus auf Maschinenausfallzeiten in industriellen Anlagen, verstärkt durch steigende Energiekosten und unvorhersehbare Lieferketten, hat sich Instandhaltung als ein wichtiges Handlungsfeld für zahlreiche Branchen entwickelt. Folglich fokussieren sich Unternehmen zunehmend auf die Optimierung von Instandhaltungsprozessen durch die Anwendung innovativer Technologien und Konzepte wie Künstliche Intelligenz (KI), Big Data und das Internet of Things (IoT) im Rahmen von Industrie 4.0. Unter den KI-Technologien haben transformerbasierte Large Language Models (LLMs) in den letzten Jahren große Aufmerksamkeit erhalten. Ihr exzellentes Sprachverarbeitungsvermögen in verschiedenen textbasierten Aufgaben übertrifft das von State-of-the-Art Systemen, was zu ihrer weit verbreiteten Nutzung als KI-Tools, virtuelle Agenten und Chatbots in verschiedenen Bereichen geführt hat. Allerdings fehlen industrielle Instandhaltungsanwendungen, die solche LLMs einsetzen. Diese Arbeit schließt somit diese Lücke, indem sie das Potenzial von LLM-basierten Chatbots in der industriellen Instandhaltung untersucht, mit dem Ziel, Instandhaltungsarbeiten zu verbessern und zu unterstützen.

Zunächst führt diese Arbeit eine umfassende Literaturanalyse durch, um vortrainierte LLMs zu identifizieren, die am besten für Feintuning und anschließende Integration in ein Chatbot-Framework geeignet sind. Anschließend stellt diese Arbeit ein auf die Instandhaltung feingetuntes LLM, unter Nutzung des BERT[1]-basierten TaPas[2]-Modells, für tabellenbasierte Frage-Antwort-Aufgaben vor. Trotz des begrenzten Datensatzes führt der Feintuningprozess zu einer erheblichen Steigerung des Exact Match Werts um bis zu 70% im Vergleich zu nicht feingetunten Modellen. Um das Problem der begrenzten Datenmenge für das Training zu adressieren, werden Datenaugmentationsmethoden wie wortbasierte Paraphrasierung und zeichenbasierte Rauschverfahren eingesetzt. Spezifische Herausforderungen der Instandhaltung werden identifiziert und ein domänenspezifischer Augmentationsansatz wird vorgeschlagen. Dieser bewahrt die Semantik und den textuellen Kontext, während er gleichzeitig die Gesamtdatenvielfalt erhöht.

Abschließend zeigt die Arbeit wie feingetunte LLMs in der Instandhaltung genutzt werden können. Das feingetunte TaPas-Modell wird in ein Chatbot-Framework integriert und dient als Frage-Antwort-System für Instandhaltungsprozesse. Der Chatbot erzielt in einer PSSUQ[3]-Nutzerfreundlichkeitsstudie eine Gesamtnote von 1,77, was auf ein positive Nutzererlebnis von ungeschulten Anwendern deutet. Dieses Ergebnis macht den LLM-basierten Instandhaltungschatbot zu einem

[1] Bidirectional Encoder Representations from Transformers
[2] Table Parsing
[3] Post-study System Usability Questionnaire

vielversprechenden Tool im industriellen Instandhaltungssektor mit der die betriebliche Effizienz in verschiedenen Branchen verbessert werden kann. Darüber hinaus könnten künftige Arbeiten den Anwendungsbereich des Chatbots auf Aufgaben wie die Wartungsplanung ausweiten und so die Grundlage für eine effektive Integration in Instandhaltungsmanagementsysteme schaffen.

Abstract

In response to the increased focus on machine downtime in industrial plants, intensified by rising energy costs and unpredictable supply chains, maintenance has emerged as an important field of action for numerous industries. Consequently, companies are increasingly focusing on optimizing maintenance processes by leveraging novel technologies and concepts such as artificial intelligence (AI), Big Data, and the Internet of Things (IoT) within the realm of Industry 4.0. Among AI technologies, transformer-based large language models (LLMs) have received remarkable attention in recent years. Their exceptional performance in natural language processing (NLP) tasks surpasses state-of-the-art systems, leading to the widespread adoption of AI tools, virtual agents, and chatbots in diverse sectors. However, industrial maintenance solutions utilizing such LLMs are missing. This master thesis bridges this gap by investigating the potential of LLM-based chatbots in industrial maintenance, aiming to enhance and support maintenance operations.

This work conducts an extensive literature analysis to identify pre-trained LLMs best suited for fine-tuning and subsequent integration into chatbot frameworks, highlighting popular models and downstream tasks. Subsequently, this work introduces an LLM fine-tuned for maintenance, using the BERT[4]-based TaPas[5] model, tailored for table question-answering tasks. Despite working with a limited dataset, the responses' exact match score for the fine-tuned solution increases substantially, reaching up to 70% compared to non-fine-tuned models. Addressing the challenge of limited dataset size, state-of-the-art NLP data augmentation methods, such as word-based paraphrasing and character-based noising techniques, are employed. Specific challenges unique to industrial maintenance are identified, and a domain-specific augmentation approach is proposed. This approach preserves semantic meaning and textual context while enhancing overall data diversity.

Finally, this work demonstrates how to leverage fine-tuned LLMs in industrial maintenance. The fine-tuned TaPas model is integrated into a chatbot framework, serving as a question-answering system supporting industrial maintenance processes. A PSSUQ[6] usability study yields an overall score of 1.77, indicating a positive user experience, particularly for untrained users. This outcome positions the LLM-based industrial maintenance chatbot as a promising tool for enhancing operational efficiency across various industries in the industrial maintenance sector. Additionally, future works could extend the chatbot's scope to tasks such as maintenance planning, laying the foundation for practical integration into maintenance management systems.

[4] Bidirectional Encoder Representations from Transformers
[5] Table Parsing
[6] Post-study System Usability Questionnaire

Table of Contents

1 Introduction

This chapter outlines the motivation for the following research. Subsequently, it presents the problem statement to be addressed, along with the derived objectives and the research questions. Finally, the research methodology and the structure of the thesis are described.

1.1 Motivation and Problem Definition

The risk of global pandemics, surging energy prices, and unpredictable supply chains increased the focus on mitigating machine downtime in industrial plant operations. This led the maintenance field to develop into a driving field of action for many industries over the last years (Cervo et al., 2023). The advent of the 4th industrial revolution (Industry 4.0) has provided new possibilities, enabling the integration of digitalization into industrial maintenance. This involves technologies like the industrial IoT and Augmented Reality (AR). Industrial IoT connects distributed physical objects over the internet and enables real-time monitoring of machines and equipment. On the other hand, maintenance concepts are extended through AR, which offers support during maintenance tasks and provides step-by-step guidance for diagnostics, inspection, and training operations (Silvestri et al., 2020). Notably, companies are also increasingly investing in chatbot technology based on the capabilities of the latest generative AI technology, exploiting their ability to deliver services at a reduced cost compared to human workers. However, the recent focus has predominantly centered on sectors such as customer services, healthcare, and personal assistance (Chui et al., 2022). As a result, domain-specific solutions in the fields of repair and service operations, maintenance, and asset management within the manufacturing industry are missing.

Historically, chatbots showed limited flexibility as their output (answers) relied on matching inputs (questions) with a predefined knowledge base, which is often hand-coded (Coli et al., 2020). This limitation restricted the effective utilization of chatbots as virtual assistants in industrial maintenance, a field heavily reliant on processing explicit and implicit knowledge. This includes unstructured text documents, such as maintenance logs, machine manuals, and non-documented knowledge of personnel, respectively.

Significant advancements have been made in the fields of NLP, natural language understanding (NLU), and generation (NLG) in recent years, driven by the development of novel transformer networks (Vaswani et al., 2017). These networks, trained on extensive textual data, have led to the creation of large pre-trained language models that demonstrate performance comparable to state-of-the-art fine-tuned systems (Brown et al., 2020; Chowdhery et al., 2022; Devlin et al., 2018; Hoffmann et

al., 2022). Despite their advancements, these systems are not infallible and are susceptible to errors, specifically the generation of ambiguous or made-up answers. Further, as components of socio-technical systems, when integrated into virtual assistants, they are required to comply with ethical guidelines and safety standards. In response, comprehensive regulatory frameworks, such as the EU AI Act proposed in April 2021, aim to regulate such systems. The EU AI Act seeks to ensure the proper functioning of the single market within the European Union by establishing conditions for developing and using trustworthy AI systems. This comprises emphasizing safety, transparency, and traceability in existing and newly introduced AI systems (ERPS, 2023).

Considering these circumstances, the extent to which the latest pre-trained models can serve as knowledge-sharing tools in maintenance management and offer support for human workers when integrated into chatbots, addressing task-related queries, should be explored. While numerous studies compare different LLMs concerning their task-specific performance (Lipenkova, 2022), the optimal open-source pre-trained models for training (fine-tuning) on domain-specific data remain unexamined. This represents problem 1 (P1) of this thesis. Further, in the context of impending regulations such as the EU AI Act, it is crucial to consider model aspects like answer accuracy, transparency, and reliability. Even though LLMs continuously demonstrate their ability to complete NLP tasks like question answering (QA), translation, and cloze completion, they still lack information that falls outside the scope of their pre-training data (OpenAI, 2023). This limitation is particularly relevant to industrial maintenance data. Consequently, there is a gap in the availability of LLMs tailored explicitly for industrial maintenance applications (P2). It is worth noting that fine-tuning necessitates several hundred to thousands of high-quality examples curated by human experts (OpenAI, 2023). Generating such an extensive dataset would be very time and cost-intensive for industrial maintenance applications, limiting the currently available training data in this field (P3).

Considering the above discussion, this thesis deals explicitly with three main problems:

- P1: It is unexamined which open-source pre-trained LLMs are best suited for fine-tuning on domain-specific data.
- P2: LLMs that are specifically trained for industrial maintenance applications are missing.
- P3: Labelled industrial maintenance training data for fine-tuning LLMs is limited.

1.2 Research Objectives

This thesis investigates the feasibility and constraints of implementing LLMs in chatbots tailored explicitly for industrial maintenance applications. The study is divided into a theoretical section, exploring the fields of industrial maintenance, NLP, and

transformer networks, and a practical section wherein an LLM-based maintenance chatbot is developed and assessed.

A systematic literature review on LLMs is conducted to achieve this objective. It determines which current pre-trained models are publicly available. Differences in training data, network architecture, and pre-training objectives are identified and analyzed. Subsequently, their potential for fine-tuning on domain-specific data is systematically evaluated and compared. As a result, it sheds light on their applicability to industrial maintenance chatbots, clarifying P1 and obtaining objective 1 (O1).

Based on these results, the best-suited pre-trained LLM shall be fine-tuned on industrial maintenance logs and eventually implemented as a chatbot. Additionally, different model training and model evaluation methods shall be analyzed. This creates an LLM specifically trained for maintenance applications, overcoming P2 and achieving O2.

The amount of training data has a high impact on the model's performance and its generalization capability. Therefore, text data augmentation is discussed in the context of industrial maintenance to generate additional training data (P3), and different augmentation methods like random swapping, insertion, and synonym replacement are analyzed. With those findings, a concept for text data augmentation in the industrial maintenance domain is derived, and O3 is attained.

The main focus of this work is the development and demonstration of an LLM-based chatbot, which can provide task-specific information during maintenance processes. The work identifies which pre-trained models are best suited for domain-specific fine-tuning, how LLMs can be fine-tuned on maintenance logs, and whether data augmentation can improve data diversity and model performance. Finally, the usability of the chatbot is evaluated through the Post-Study Usability Questionnaire (PSSUQ) (J. R. Lewis, 1995) to assess its applicability in the maintenance industry. This leads to the main research question that is answered in this thesis: "To what extent can LLM-based chatbots effectively be used in industrial maintenance applications?"

The sub-research questions (RQ), which are derived from P1-3, are:

- RQ1: Which open-source pre-trained LLMs are best suited for fine-tuning on domain-specific data?
- RQ2: To what extent is it possible to fine-tune an LLM with maintenance logs and implement it as a chatbot?
- RQ3: What are domain-specific data augmentation approaches, and how can they enhance the availability and diversity of labeled training data for fine-tuning LLMs?

The objectives of this thesis are:

- O1: Providing an overview of state-of-the-art LLMs and their potential for domain-specific training.
- O2: Fine-tuning an LLM with industrial maintenance data and implementing it as a chatbot.
- O3: Conception of a data augmentation approach in the context of industrial maintenance for improving overall data diversity and model robustness.

1.3 Methodology and Thesis Structure

The research of this thesis follows the design science research process (DSRP) model proposed by Peffers (2006). It is a nominal process including six activities for conducting research in information systems (IS), ultimately providing a guideline for producing and presenting design science research in IS. The model offers several entry points for research, as seen in Figure 1 (Peffers, 2006). The development of an LLM-based chatbot for maintenance applications, which is the central part of this thesis, can be viewed as an objective-generated solution in the scope of the proposed DSRP model. Research starts by looking at the results of an existing artifact (traditional chatbots) and trying to improve them by introducing an alternative artifact (e.g., LLM-based chatbots).

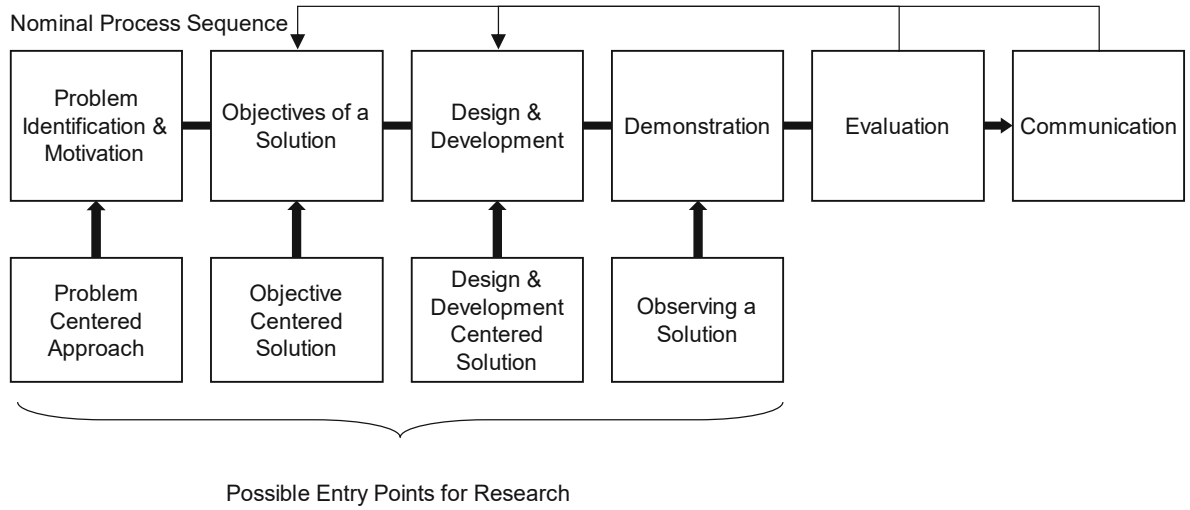


Figure 1 | DSRP model (Peffers, 2006)

In the subsequent design and development activity, the actual artifact – a prototype of a maintenance chatbot - with its desired functionality is created. First, chatbot requirements for the industrial maintenance sector are defined and an LLM is selected for fine-tuning. Subsequently, the model is fine-tuned on real-world maintenance data and performance improvements are measured using task-specific metrics. Finally, the LLM is implemented into a chatbot framework. During the industrial chatbot development, data augmentation approaches to extend sparse domain-specific training data sets will be researched and evaluated. The results should introduce a

domain-specific data augmentation approach in the context of industrial maintenance to enhance overall data diversity and model robustness as an additional artifact.

To achieve these goals, extensive knowledge of the underlying subject and theory is required (Peffers, 2006) and is attained through detailed literature research regarding industrial maintenance and NLP. Furthermore, current LLMs are analyzed and categorized in a systematic literature review regarding their differences in training data, network architecture, pre-training objectives, and domain-specific fine-tuning potential. Its methodology is based on the strategy of Zonta et al. (2020) and Peres et al. (2020). Google Scholar, Scopus, and IEEE are defined as databases for which a search string was constructed.

Finally, the last three activities established by Peffers (2006) are addressed in this thesis. The efficacy of the artifact in solving the problem is demonstrated using maintenance queries. The usability of the chatbot is evaluated by conducting usability studies according to the PSSUQ, a 16-item instrument for assessing user satisfaction with digital systems (J. R. Lewis, 1995). Iterations between steps 3 and 5 are performed to achieve satisfying results. The last activity, communication, is represented by this thesis itself, using the proposed DSRP model as its primary structure.

Table 1 presents the comprehensive framework of this thesis, outlining the topics and objectives covered in each chapter. Chapters 1 and 2 serve as introduction segments, describing this study's motivations, necessity, research questions, and objectives. Chapter 2 establishes the theoretical foundation of this work, exploring scientific domains such as "Maintenance in production systems," "Natural language processing," and "Transformer networks." Chapters 3, 4, and 5 comprise the practical aspects of this thesis. Chapter 3 systematically identifies the most suitable LLMs for fine-tuning on domain-specific data through a systematic literature review. Building upon these findings, Chapter 4 proposes an industrial maintenance LLM by fine-tuning it with domain-specific training data, employing data augmentation techniques to enhance data diversity and the model's robustness. This chapter also details the integration of the fine-tuned model into a chatbot framework. Chapter 5 assesses the question-answering accuracy of the fine-tuned model and the usability of the developed maintenance chatbot.

Finally, Chapters 6 and 7 provide a summary of this work. Chapter 6 addresses the limitations of the methods and approaches employed, while Chapter 7 consolidates the results and outlines future research directions.

Table 1 | Structure of this thesis

Chapters	Topics	Objectives
1. Introduction	Introduction to the topic; Derivation of research questions and objectives; Definition of a research methodology	Highlighting the necessity of this work and outlining the research problems and objectives
2. Theoretical Background	Explanation of scientific areas such as „Maintenance in production systems, “Natural language processing, “and „Transformer networks.“	Providing a comprehensive understanding of the underlying topics covered in this thesis
3. State-of-the-art Domain-specific Fine-tuning Strategies of Pre-trained LLMs	Outlining state-of-the-art fine-tuning methods, including models and downstream tasks, by conducting a systematic literature review	Identifying the best-suited LLMs for domain-specific fine-tuning
4. Development of a Maintenance Chatbot	Description of the domain-specific fine-tuning process (including the data augmentation process of maintenance data) of an LLM and its integration into a chatbot framework	Proposal of an industrial maintenance chatbot prototype that leverages a domain-specific LLM
5. Evaluation	Description of the QA accuracy and chatbot usability evaluation process with subsequent presentation of its results	Identifying the advantages and disadvantages of domain-specific fine-tuning of LLMs as well as LLM-based industrial maintenance chatbots
6. Limitations	Outlining limitations of the methods and approaches used in this work	Identifying possible improvements that could be included in future works
7. Conclusion and Outlook	Conclusion of results and proposition of future work directions	Highlighting the achievements of this work and pathways for future research

2 Theoretical Background

This chapter provides the theoretical foundation for this thesis, as outlined in Section 1.3. Initially, it delves into the domain of industrial maintenance within production systems, defining key terminology and foundational principles (cf. Section 2.1). Subsequently, the chapter explores industrial maintenance in the realm of Industry 4.0, delving into the integration of digital assistants in this context. Second, the field of NLP and its application in digital assistants will be detailed (cf. Section 2.2). The section will conclude by describing transformer-based LLMs for solving common NLP tasks (cf. Section 2.3).

2.1 Maintenance in Production Systems

The overarching objective of maintenance is to ensure the availability, reliability, maintainability, and safety of plants. This task has gained significant prominence across various industries, particularly with the growing complexity and automation of production processes (Matyas, 2022a). Thomas & Weiss (2020) conducted a survey among 85 US manufacturers, finding that direct and indirect maintenance costs among the participants totaled 74.5 billion US dollars in 2016. Therefore, implementing a holistic approach encompassing direct maintenance expenses and plant downtime costs holds substantial saving potential. Such an approach aims to minimize direct costs by enhancing the availability of machines and plants while sustaining consistent output levels (Matyas, 2022a).

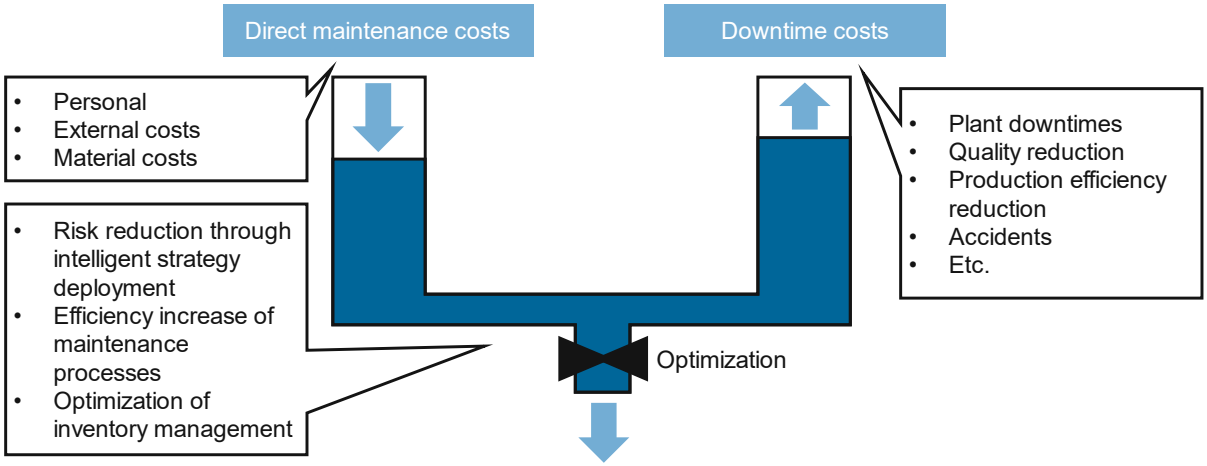


Figure 2 | Holistic view on industrial maintenance costs (Matyas, 2022a)

This correlation is visualized in Figure 2. It can be seen that reducing direct maintenance costs by trimming the maintenance budget results in increased machine downtime and associated costs. The optimal reduction of both direct maintenance expenses and consequent downtime costs, therefore, requires optimizing the maintenance process itself (Matyas, 2022a). This optimization can be realized by

implementing advanced maintenance strategies, such as predictive maintenance, improving management of maintenance personnel, and improving the sharing of maintenance knowledge (Cervo et al., 2023).

Consequently, multiple maintenance strategies are available and employed throughout the industry today. However, their costs and effectiveness vary, such that poorly implemented strategies will have little success in achieving the business objective, such as reducing plant downtime and repair costs (Deighton, 2016b). Thus, clear definitions of underlying concepts and terminologies must be made first.

2.1.1 Definition of Terms

The term maintenance is defined in the European Standard EN 13306. It is the

"combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function" (BSI Standards Publication, 2018, p. 8).

In this context, an item refers to a device, system, or sub-system that can be considered individually. At the same time, its life cycle is defined as the stages from conception to disposal (BSI Standards Publication, 2018).

The German Standard DIN 31051 goes further and categorizes maintenance actions into four main activities, as displayed in Figure 3 (Matyas, 2022a):

- **Service:** Service measures are crucial for preserving the initial state of technical equipment. These measures reduce the rate of wear and tear, leading to an extended operational lifespan. The sub-measures of service involve cleaning, conserving, readjusting, lubricating, supplementing, and replacing. These actions are essential for sustaining the efficiency and longevity of machinery and equipment.
- **Inspection:** The inspection process requires consistent assessment of the item's condition under constant operational and environmental conditions. The findings should maintain standards and tolerances, aligning with their initial state. In summary, inspection involves identifying the condition of technical equipment, assessing it, evaluating the state information, determining wear causes, conducting error analysis, and initiating further actions based on the evaluated condition.
- **Repair:** The repair process is divided into two sub-measures, which are repair through modification and repair through replacement.
- **Rebuilding:** Maintenance includes recognizing technical improvements based on economic evaluation. Thus, improvement enhances operational reliability by eliminating weaknesses, while modifying a unit to fulfill a different function is not

considered maintenance. Fault analysis assesses if improvement is feasible, considering economic viability.

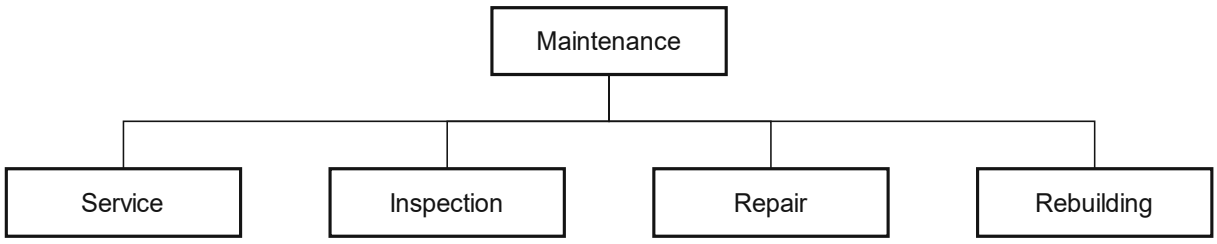


Figure 3 | Categorization of maintenance activities (DIN, 2012)

An exemplary wear reserve of a unit can be seen in Figure 4. It starts at 100% and degrades gradually over time. However, when the wear level reaches a specific threshold, a maintenance action is initiated to restore the unit's functionality.

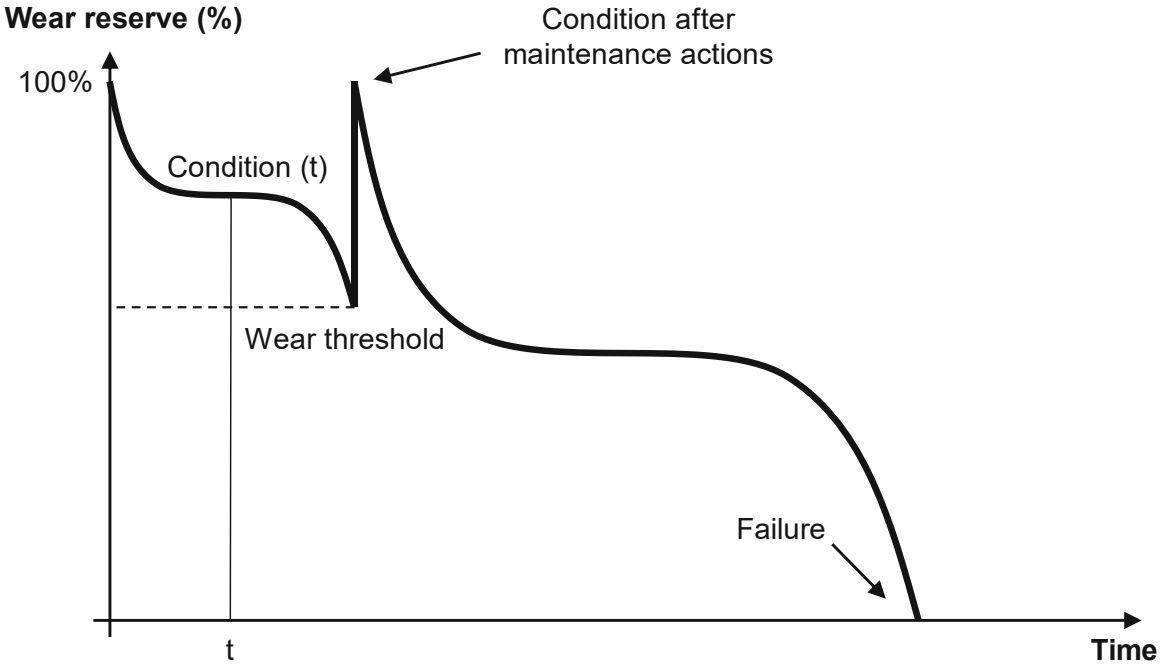


Figure 4 | Exemplary wear reserve trajectory of equipment (Matyas, 2022a)

Specific maintenance concepts dictate the timing of a maintenance action within the unit's life cycle. These concepts fall into two primary categories, as outlined by Deighton (2016):

- **Reactive maintenance:** This type involves responding to equipment failures and unplanned downtimes. It refers to breakdown and corrective maintenance, addressing issues only after they occur.
- **Proactive maintenance:** Proactive maintenance is conducted before urgent issues occur. It aims to identify and correct conditions that could lead to equipment degradation, thus preventing costly repairs and downtime. Proactive maintenance can be further divided into:

- **Preventive maintenance:** Activities are performed based on a predefined schedule, including inspections, replacements, and servicing tasks to ensure that equipment remains functional.
- **Predictive maintenance:** This systematic approach involves assessing the equipment condition to determine if it is nearing failure, allowing for timely repairs or replacements.

While reactive maintenance strategies have been popular since the 1940s because of their low implementation costs, they cannot be planned and eventually result in the unavailability of a system after its failure (Nowakowski et al., 2019). Therefore, more sophisticated maintenance concepts have emerged in the era of digitalization and the context of Industry 4.0.

2.1.2 Industrial Maintenance in the Context of Industry 4.0

The industrial revolutions represent significant technological advancements in the past and future. The fourth and latest industrial revolution is also represented by the term Industry 4.0 and is characterized by interconnected systems using the latest internet technology (Schnelzer, 2019). There are various definitions in the scientific literature, but Roth (2016) defines it as:

"Industry 4.0 encompasses the networking of all human and machine actors along the entire value chain as well as the digitization and real-time evaluation of all relevant information with the aim of making value creation processes more transparent and efficient in order to optimize customer benefits with intelligent products and services." (Roth, 2016, p. 6)

It combines production and automation technology to achieve a new level of organization and control across the entire product lifecycle. The goal is significant flexibility, improved value creation, and product individualization (Roth, 2016). Thus, according to (Roth, 2016), companies embracing this revolution aim for:

- High product individualization
- Highly flexible and efficient production
- Extensive integration of customers and partners into business
- Integration of production with high-value services, resulting in hybrid products

Eventually, these goals and technological advancements lead to more interconnected and complex production systems, which have significantly impacted current maintenance strategies (Matyas, 2022a). This trend resulted in developing data-driven maintenance strategies, specifically predictive and prescriptive approaches. They are pivotal innovations within the digitalization of industrial maintenance because they enable targeted influence on processes and decisions within an industrial environment despite the increased complexity of production systems. This results in enhanced

planning reliability, increased plant productivity, and reduced operational costs (Glawar, Nemeth, et al., 2022).

Organizations that commit to predictive maintenance strategies can monitor the condition of critical equipment and can predict trends, patterns, and correlations by mathematical models for anticipating failures in advance (Nair, 2023; Zonta et al., 2020). Its importance is underlined by its online global search interest, as seen in Figure 5. Its interest has been rising for the last 12 years and has grown nearly threefold since 2017, outgrowing other maintenance trends (Brügge, 2023).

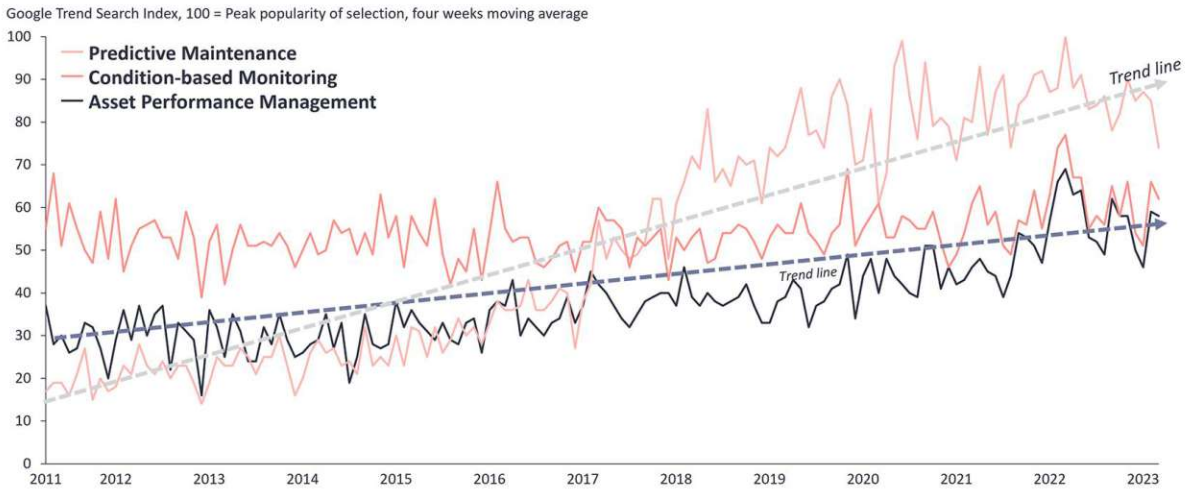


Figure 5 | Global search interest for predictive maintenance (Brügge, 2023)

While predictive maintenance answers the question “What will happen when?“ prescriptive maintenance builds upon this question with the following question: “How should it happen?” (Glawar, Nemeth, et al., 2022). Ansari et al. (2019) elaborate on this in their proposed prescriptive maintenance model (PriMa), which is comprised of four layers (data management, a predictive data analytic toolbox, a recommender and decision support dashboard as well as an overarching layer for semantic-based learning and reasoning). Together, they enable the generation of decision-support measures and recommendations for the improvement and optimization of incoming maintenance plans. Its practical potential was verified in an industrial environment, which revealed significant improvements in the reduction of downtime, especially for load-dependent behavior of parts (Ansari et al., 2019).

According to Ansari & Glawar (2022), both strategies fall within the scope of the term “Knowledge-Based Maintenance” (KBM). It is a holistic concept encompassing several maintenance strategies and approaches, including predictive and prescriptive maintenance. It primarily aims to reduce maintenance costs by taking a holistic rather than partial view of all influencing components and gathering extensive knowledge. In the scope of this concept, maintenance is considered a non-isolated sub-domain of production systems that has a significant influence on organizational value creation (Ansari & Glawar, 2022).

Consequently, today's maintenance concepts go beyond maintaining operations and ensuring availability (Matyas, 2022a). This vision is pursued by Glawar, Ansari, et al. (2022) in their proposed Maintenance-Free Factory (M2F) approach, which explores unexhausted potentials for transforming maintenance into an enabler instead of a cost-driven system in manufacturing. In this context, it aims to reduce the proportion of technical and technological malfunctions through the effective use of data-driven and AI-enhanced methods (Glawar, Ansari, et al., 2022). Additionally, it aims to decouple maintenance activities from the production process by optimizing flexible shift planning between previous production and maintenance planning procedures. As a result, this concept requires a strategic rethinking of maintenance and production management with organizational changes on strategic, tactical, and operative levels (Glawar, Ansari, et al., 2022).

2.1.3 Digital Assistants in the Industrial Maintenance Industry

Digital assistants gain increased attention through digitalization across all industries because they offer targeted support for workers during various tasks and improve their efficiency and the quality of their work (Apt et al., 2018). Improved maintenance workflows are one lever to increase productivity and reduce costs in the industrial maintenance industry (Matyas, 2022a). In this context, digital assistants can provide domain-specific knowledge to assist workers on the production floor and information for administrative staff for improved decision-making. Consequently, digital assistants must be considered an essential part of improving productivity and efficiency in modern factories (Pereira et al., 2023).

Wellsandt et al. (2021) defined digital assistants as socio-technical systems with individual user considerations, goals, tasks, and interactive technology. These systems include various components to fulfill specific functions, defining them as an application class. General-purpose assistants aid in tasks like information retrieval, calendar management, communication, and device control. Special-purpose assistants provide support for specific tasks. Adaptive assistants learn from interactions, enhancing language skills and user experience (Wellsandt et al., 2021).

Knote et al. (2019) performed a systematic literature review on 115 digital assistants and further categorized them into five distinctive groups:

- **Adaptive voice/vision assistants** primarily use speech interaction, often augmented with visual features. These assistants excel in NLU and perform diverse tasks based on user requests. They evolve over time, improving service quality through adaptive knowledge models. These assistants include popular examples, embodied with computer-generated voices, such as Amazon's Alexa™, Apple's Siri®, Google's Assistant™, Microsoft's Cortana™, and Samsung's Bixby™.

- **Chatbot assistants** use text chat interaction to provide assistance. These chatbots are employed in support services, answering frequently asked questions (FAQ) and guiding users through processes. They interpret natural language, operate via text-based interfaces, and may include images or videos. Unlike the first group, these chatbots focus on presenting information on specific domain knowledge rather than executing third-party services.
- **Embodied virtual assistants** are the largest category of assistants and consist of virtual assistants designed for specific tasks. They have speech and visual capabilities, providing human-like interaction through screen-based interfaces. Users typically have limited control over them, with some systems adapting to user preferences. These assistants aim to replicate human-to-human activities.
- **Passive pervasive assistants** minimize user interactions while collecting sensor data. These assistants initiate interactions and assist primarily through screen output. Most of them perform actions based on sensed triggers autonomously. They aim to seamlessly integrate into the user's environment, improving experiences without actively disturbing the user.
- **Natural conversation assistants** focus on speech interaction to mimic human-to-human natural language communication. These assistants have advanced speech recognition and language understanding capabilities, enabling them to comprehend complex speech patterns. The goal is to provide a natural interaction experience.

In the context of industrial maintenance applicability, 21% of experts give the implementation of “mobile (digital) assistance systems” and “real-time decision support for maintenance planners” high relevance (Glawar, Nemeth, et al., 2022), as seen in Figure 6.

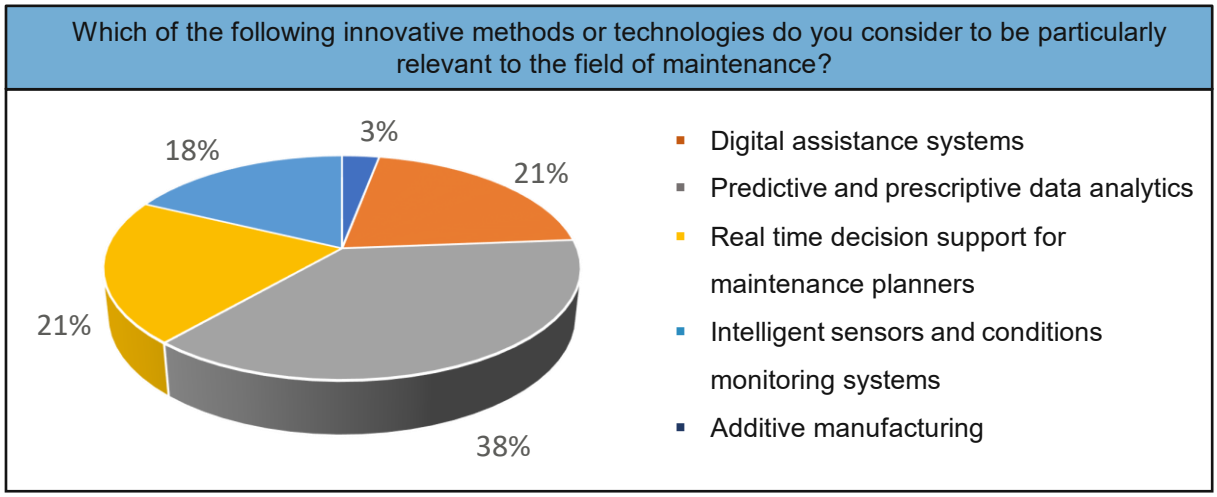


Figure 6 | Survey results, indicating innovate trends and applications in the field of industrial maintenance (Glawar et al., 2022)

They provide the ability to efficiently share knowledge that already exists in an organization and help new employees become more productive more quickly by providing recorded procedures used for critical tasks (Cervo et al., 2023). This helps

tackle the problem of the increasing complexity of machinery in the maintenance domain, leading to equally complex repair and maintenance procedures, which require increased expertise among maintenance personnel (Glawar, Nemeth, et al., 2022). Thus, in the era of digitalization, digitizing maintenance expertise and increasing its accessibility through digital assistance systems using advanced computer algorithms should be considered.

2.2 Natural Language Processing

To allow digital assistants, like chatbots, to interact with their users, they need to process and understand natural language. This process is called "Natural language processing" (NLP) and is defined as:

"…a field that addresses various ways in which computers can deal with natural – that is, human – language." (Kochmar, 2022, p. 1)

NLP itself can be split into two main parts following Khurana et al. (2023):

- **Natural language understanding** (NLU) enables machines to comprehend and analyze human language by extracting concepts, entities, emotions, and keywords.
- **Natural language generation** (NLG) involves creating meaningful phrases, sentences, and paragraphs from an internal representation.

How NLP relates to fields like AI, machine learning (ML), and deep learning (DL), which are often used in a similar context, is shown in Figure 7.

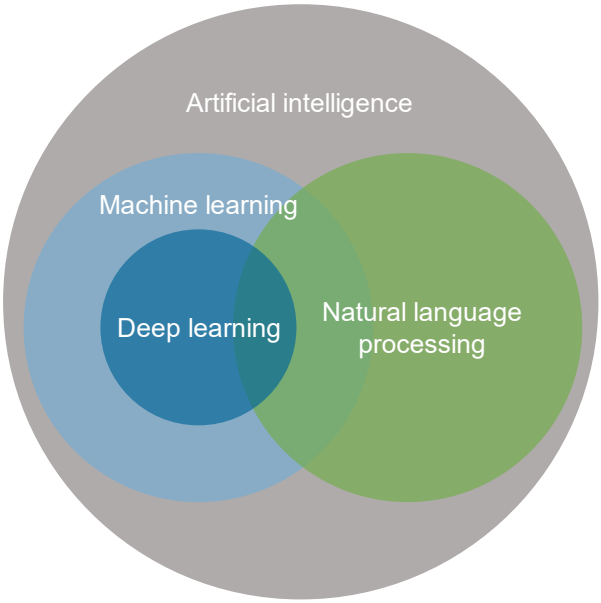


Figure 7 | AI is the overarching domain for closely related concepts such as NLP, machine, and deep learning (Hagiwara, 2021)

AI serves as the overarching domain, including various subfields like ML, NLP, computer vision, and speech recognition (Hagiwara, 2021). ML is a subset of AI and enables computer program performance to improve with experience using supervised, unsupervised, and reinforcement learning methods (Janiesch et al., 2021). DL, a subfield of ML, employs deep neural networks with improved learning capabilities (Janiesch et al., 2021). Modern NLP heavily relies on ML and DL techniques. However, specific traditional NLP methods, like word counting and text similarity measurements, are essential building blocks for ML-based models but are not always categorized as ML (Hagiwara, 2021).

The historical evolution of NLP approaches is depicted in Figure 8, starting in 1950. Early NLP approaches relied on rule-based and template-based systems, where linguists and language experts hardcoded patterns and rules for tasks like translation and information extraction. These methods utilized expert knowledge. However, human language's diversity and ambiguity posed challenges, limiting the rule-based method's generalization ability (Kochmar, 2022). In the 1980s, the introduction of statistical approaches and ML algorithms marked significant progress in NLP tasks, leveraging language data and statistics to improve performance (Kochmar, 2022). While rule-based approaches in NLP rely on precise but inflexible rules, statistical approaches learn from data without assumptions, allowing prediction flexibility (Kochmar, 2022). However, statistical methods require large, high-quality datasets to perform well. Eventually, with the continuous development of algorithms and the growth in available data, novel approaches capable of learning efficiently from vast datasets were required. In the 2010s, advancements in computer hardware enabled the adoption of powerful DL techniques, which gained significant prominence in the last couple of years (Kochmar, 2022).

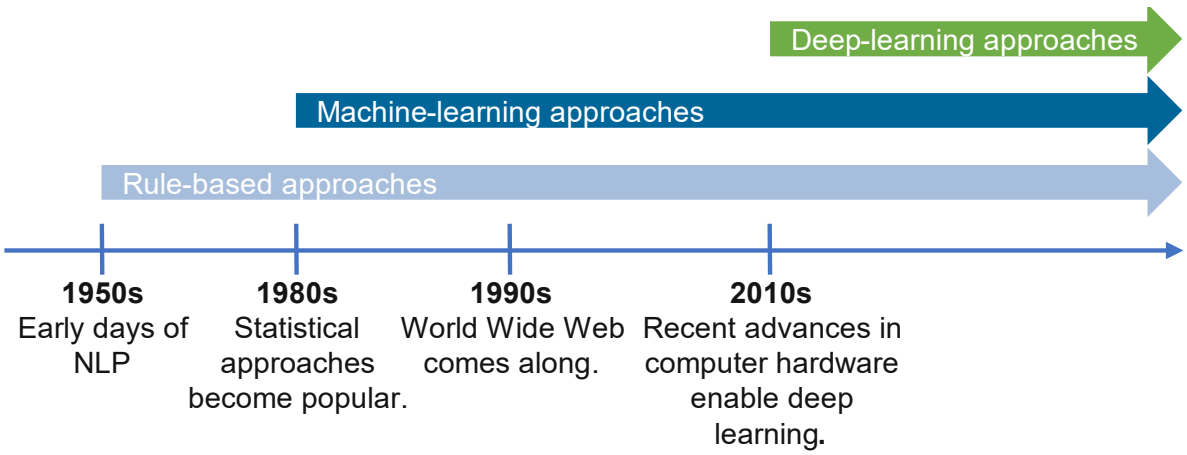


Figure 8 | Different approaches that emerged throughout the lifespan of NLP (Kochmar, 2022)

In the context of industrial applicability, Brundage et al. (2021) found that significant problems (e.g., incorrect tokenization and stop word removal) can arise if common NLP methods are applied to technical text. Thus, they additionally proposed an extension to NLP, "Technical Language Processing" (TLP), which is defined as:

"…a human-in-the-loop, iterative approach to tailor NLP tools to engineering data." (Brundage et al., 2021, p. 44)

Thus, they see TLP as a socio-technical system maintained by individuals rather than as an algorithmic pipeline (Dima et al., 2021). It aims to utilize domain-specific taxonomies and data dictionaries to ensure that industrial tools understand and process all relevant technical terms correctly (Flare, 2021). Further, it encourages the choice of the most practical techniques for developing industrial applications, achieving a thoughtful balance between analytical performance and available resources. Eventually, TLP should help mitigate the accumulation of systemic technical bias (Dima et al., 2021).

2.2.1 Approaches to Natural Language Processing

In NLP, rule-based, statistical, and DL methods are all utilized based on specific task requirements (Kochmar, 2022). Today, methods and approaches in NLP are clustered into three categories, according to Vajjala et al. (2020):

- Heuristic-based NLP
- Machine learning for NLP
- Deep learning for NLP

Heuristic and statistic-based NLP

The first heuristic-based methods are based on digitizing resources like dictionaries and thesauruses. For instance, lexicon-based sentiment analysis utilizes counts of positive and negative words to predict text sentiment (Vajjala et al., 2020). Miller et al. (1993) organized lexical information based on word meaning and created the knowledge base WordNet. It provides semantic relationships between words such as synonyms, hyponyms, and meronyms (Vajjala et al., 2020). A network representation of semantic relations among a variety of lexical concepts is displayed in Figure 9.

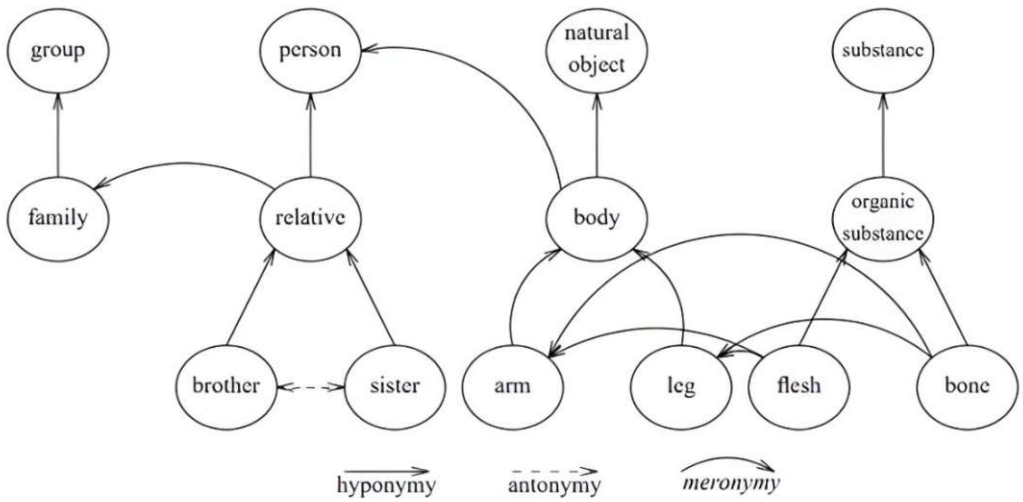


Figure 9 | Visualization of the semantic relationships incorporated in the knowledge base WordNet (Miller et al., 1993)

Regular expressions (regex) are tools for searching and manipulating text strings. A regex can be viewed as a filter that consists of a sequence of characters and metacharacters, accepting a set of strings and rejecting the rest (Hock-Chuan, 2018). Regexes enable the integration of domain knowledge into NLP systems. For instance, in customer complaints received via email, regexes can automate the identification of products that are mentioned (Vajjala et al., 2020). They are used for building deterministic rule-based frameworks like TokensRegex (Stanford NLP Group, 2013), which aid in defining regular expressions to identify text patterns and create rules. The sub-branch of probabilistic regexes introduces probabilities for matching, addressing the deterministic limitations (Vajjala et al., 2020).

Machine learning for NLP

ML techniques, including supervised methods like classification and regression, are widely used in NLP tasks such as news article categorization and stock price estimation based on social media discussions. In contrast, unsupervised clustering algorithms are mainly deployed for grouping text documents (Vajjala et al., 2020). One supervised method is the Naive Bayes classifier, which calculates class label probabilities. It uses Bayes' theorem and assumes independence among features (Chakraborty et al., 2019). This makes it suitable for text classification tasks, like news article classification based on word counts (Vajjala et al., 2020).

A support vector machine (SVM) is a binary classifier that implicitly transforms data from the original feature space to a higher-dimensional space using a kernel function. This transformation disentangles the data, making it linearly separable. The process involves mapping the input space to an alternative representation with a higher dimension, enabling the creation of a hyperplane for classification (Raaijmakers, 2020). Ansari et al. (2021) used such a method for enhanced failure detection and availability optimization in production systems. Through the training of an SVM using vectorized textual failure reports and associated downtimes, they were able to give early downtime predictions for improving industrial maintenance.

Hidden Markov models (HMM) are statistical models that relate a sequence of observations to a sequence of hidden states (Jurafsky & James H, 2023). They also make the Markov assumption that each hidden state is dependent on the previous ones (Vajjala et al., 2020). In NLP, HMMs model the hidden parts of speech beneath observable words, capturing the sequential nature of language. This works well because human language is sequential in nature, and the current word in a sentence depends on the prior words. Hence, despite their underlying complexity, HMMs are very efficient in modeling textual data (Vajjala et al., 2020).

Deep learning for NLP

In recent years, deep neural network architectures in NLP have gained prominence due to the requirement of dealing with complex, unstructured data by processing natural language. Recurrent neural networks (RNN), designed to process sequential data, are capable of understanding language flow and maintaining memory. This memory is temporal, and the information is stored and updated at every time step as the RNN processes each word in the input sentence. This aspect makes them very efficient in solving various NLP tasks such as text classification, named entity recognition, and machine translation. However, traditional RNNs are fundamentally limited by their short memory and struggle with longer contexts (Vajjala et al., 2020).

The long short-term memory networks (LSTM) address the limitations of simple RNNs by including gating operations that control the flow of historical network information into the present (Raaijmakers, 2020). The essential components of LSTM networks are seen in Figure 10. The input gate applies a weight matrix to the new input. The state cell processes the weighted input and applies a forget gate that weighs information from the previous state cell and also weighs the input using a separate matrix. Finally, the output gate selectively transmits information from the current state cell. Because of their improved memorization capabilities and their ability to focus on relevant contexts, LSTMs have largely replaced RNNs in NLP applications today (Raaijmakers, 2020).

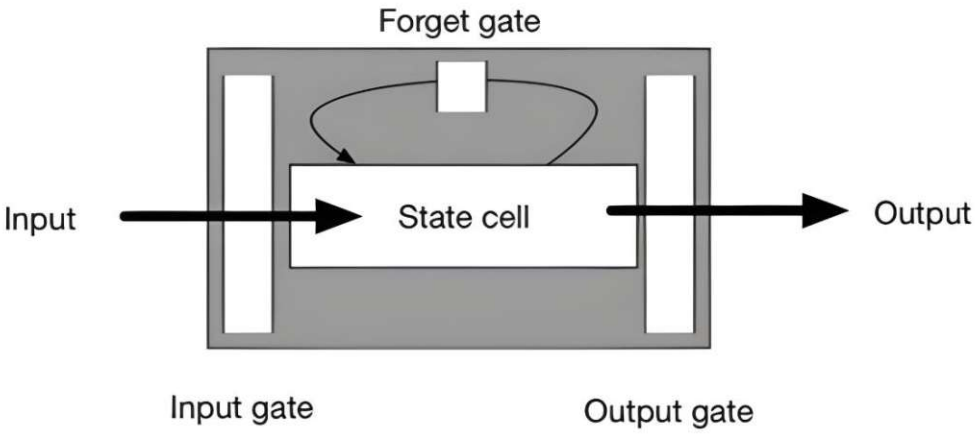


Figure 10 | Simplified architecture of an LSTM cell, showing how relevant information is maintained throughout the encoding process (Raaijmakers, 2020)

Even though LSTM networks have made significant progress in memorization capabilities and performance in common NLP tasks, transformer networks represent the forefront of DL models in the field of NLP. Over the past two years, these models have excelled in nearly all significant NLP tasks, achieving state-of-the-art performance. Unlike sequential methods, transformers do not linearly process textual context. Instead, when given a word as input, they employ a specific self-attention

mechanism to consider all words close to it, allowing each word to be represented in relation to its surrounding context (Vajjala et al., 2020).

An in-depth description of the transformer architecture, transformer-based language models, and their application for domain-specific NLP tasks, as well as the integration into chatbot frameworks, is given in Section 2.3.

2.2.2 Natural Language Processing Tasks

Several common NLP tasks, including NLU and NLG tasks, are solved in various applications today with heuristic-based, ML, and DL methods. The spectrum of such tasks is very diverse, and several tasks are intertwined with each other, such that there are sub-tasks used for higher-level tasks. Depending on the NLP task, it might require solving only NLU or NLG sub-tasks or both. This prohibits clear categorization. Therefore, this thesis defines several common high-level NLP tasks used in practical applications, which are:

- **Information retrieval:** Here, queries and documents are represented as vectors. This vector representation visualizes queries and documents in a geometrical space. In this space, objects are defined by coordinates, and their proximity determines the similarity between objects. By leveraging this geometrical space representation, the system identifies the document most relevant to the query by locating the one with the most similar content using different NLU methods (Kochmar, 2022).
- **Text classification (TC):** TC involves categorizing pieces of text into distinct classes. One notable application is spam filtering, where emails or other text types like web pages are classified as either spam or not (Hagiwara, 2021). Another essential type of text classification is sentiment analysis, which detects subjective information such as opinions, emotions, and feelings within the text using various NLU methods. Given these capabilities, sentiment analysis plays a vital role in understanding public sentiment, user reviews, and social media interactions (Hagiwara, 2021).
- **Information extraction:** This task involves extracting information from text sources, such as calendar events from emails or individuals' names in social media posts. In the literature, it is also often referred to as named entity recognition (Vajjala et al., 2020).
- **Question-answering (QA):** Various QA variants are defined based on input and output formats. Extractive QA involves extracting answers from contexts, including text, tables, or HTML. Open generative QA generates free text responses directly from given contexts. On the other hand, closed generative QA generates answers without any context provided, relying entirely on the model's generation capabilities (Hugging Face, 2023d).

- **Summarization:** Summarization condenses lengthy texts while retaining their core meaning, producing shorter text output. This method is particularly valuable for simplifying complex documents such as legislative bills, legal papers, and scientific articles. The two fundamental approaches in summarization are extractive summarization, which involves selecting and extracting important sentences directly from the source text, and abstractive summarization, which generates a summary, potentially incorporating new words not present in the source document (Hugging Face, 2023d).
- **Machine translation:** The complexity of translating sentences lies not just in word translation but in preserving sentence meaning, grammar, and tenses. This is solved using statistical ML techniques that gather parallel data from multiple sources, calculating the likelihood that language A corresponds accurately to language B or by employing artificial neural networks and DL methods (Khurana et al., 2023).
- **Text generation (TG):** TG creates natural language text from various sources. If the input data consists of text or other data types, TG can be further divided into text-to-text and data-to-text generation. For instance, dialog systems produce natural utterances based on ongoing conversations, enabling the generation of news text from events such as sports game outcomes and weather conditions (Hagiwara, 2021).

2.2.3 NLP in Conversational Assistants

In conversational agent systems, like chatbots, many NLP tasks need to be considered and resolved to allow a fluent and natural conversation with their users. A conversational assistant's general architecture is displayed in Figure 11.

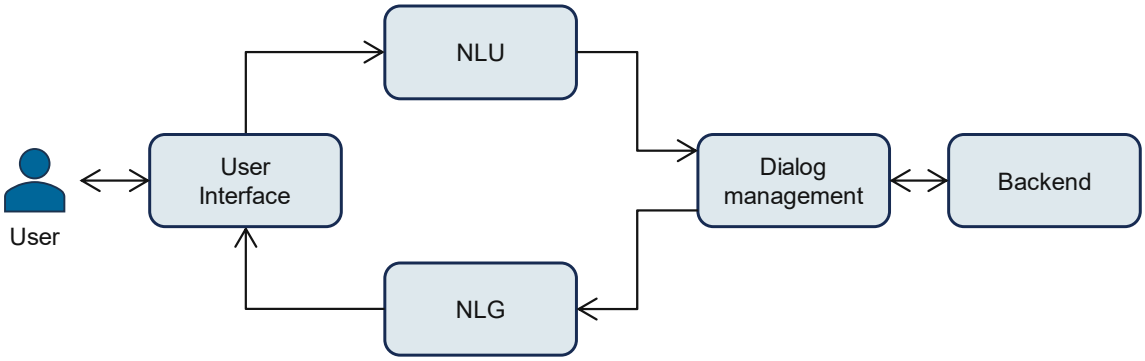


Figure 11 | A general chatbot architecture, including a user interface and NLU, NLG, and dialog management components that communicate with the chatbot's backend

In this example case, it is assumed that the assistant is text-based and communicates with the user through a common user interface. After the user's request is entered, the NLU component analyzes the text, addressing NLP tasks like sentiment detection and named entity extraction (Vajjala et al., 2020). Since users convey their intent through their requests, the NLU component's task involves understanding this intent and

executing the necessary actions. Some user inputs may include parameters, referred to as entities, which provide specific details related to these intents. Therefore, NLU is essential in extracting implicit and explicit information from the input text, laying the foundation for effective dialog management (Adamopoulou & Moussiades, 2020).

The dialog management component in chatbot systems maintains and updates the conversational information, like entities. If the component cannot collect dialog information, it actively seeks additional details from the user. Thus, according to Adamopoulou & Moussiades (2020), additional tasks of this component are:

- **Ambiguity handling**: When the intent is unclear or no input is recognized, it seeks clarification, initiates new discussions, or provides a general answer.
- **Data handling:** It stores user information, allowing the chatbot to tailor its responses based on past user's inputs.

Following intent identification, the chatbot proceeds with actions, such as information retrieval or generating user responses. Intent-related information can be retrieved through the backend using external API calls or database requests. After the relevant information is extracted, it is subsequently fed to the dialog management module and the NLG module for further processing (Adamopoulou & Moussiades, 2020).

Eventually, the NLG module crafts a human-readable response based on the strategy determined by the dialog manager. This response can be template-based or generated using language models. The generated text is then conveyed back to the user through the user interface as an answer (Vajjala et al., 2020).

2.3 Transformers for Natural Language Processing

Section 2.2.1 discussed several approaches to solving common NLP tasks, from simple rule-based to more complex DL methods. Transformers, a recent addition to DL models in NLP, have emerged as state-of-the-art models in various NLP tasks over the last couple of years. Unlike sequential approaches, transformers utilize self-attention, allowing them to consider all words in the input during encoding and decoding. This unique ability has enabled transformers to achieve state-of-the-art performance in numerous NLP tasks compared to traditional deep networks (Vajjala et al., 2020).

This chapter sheds light on the internal architecture of transformer networks, explains the basics of the self-attention mechanism, provides an overview of emerging pre-trained LLMs, and addresses the method of transfer learning in NLP.

2.3.1 Transformer Architecture

The original transformer is built upon the encoder-decoder architecture shown in Figure 12. This architecture comprises two components, as described by Tunstall et al. (2022):

- **Encoder:** It transforms an input sequence of tokens into a series of embedding vectors, often called the hidden state or context.
- **Decoder:** Leveraging the hidden state from the encoder, the decoder iteratively produces an output sequence of tokens, generating one token at each step.

The original Transformer model comprises a stack of 6 layers, where the output of each layer serves as the input for the subsequent layer until the final prediction is made. This structure includes a 6-layer encoder stack on the left and a 6-layer decoder stack on the right. Inputs enter the encoder side through an attention sub-layer and feedforward network (FFN) sub-layer, while target outputs enter the decoder side via two attention sub-layers and an FFN sub-layer (Rothman, 2021).

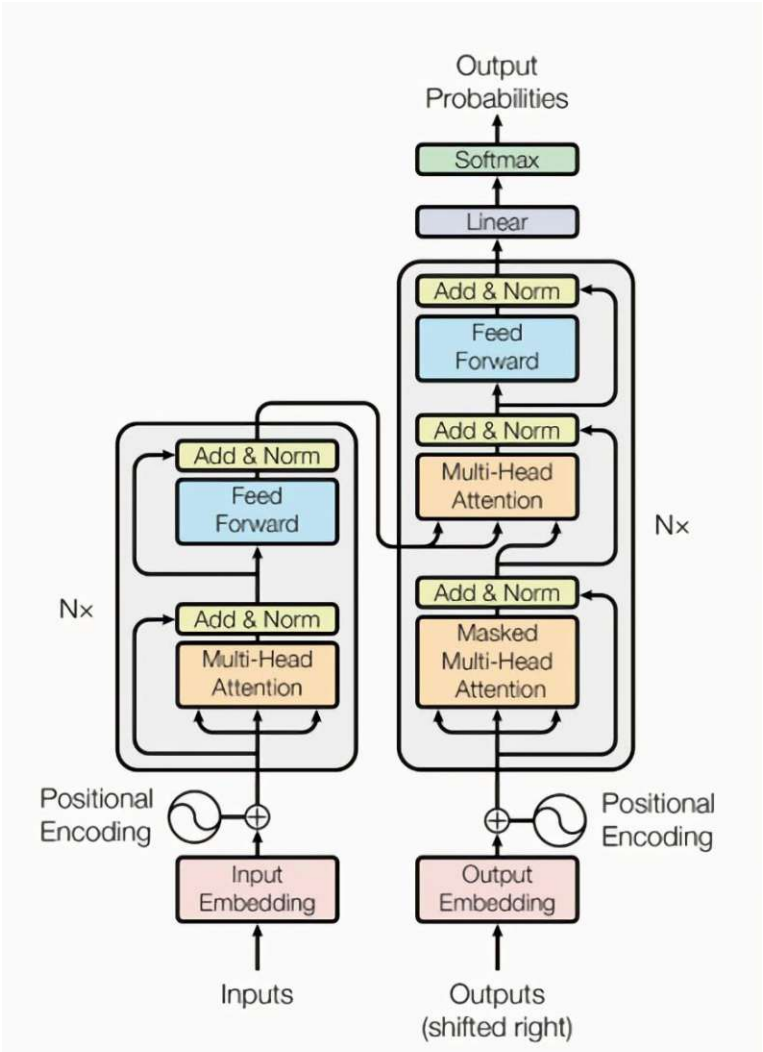


Figure 12 | The original transformer architecture, comprising an encoder and decoder stack (Vaswani et al., 2017)

Self-attention mechanism

The attention mechanism has replaced recurrence due to its ability to assess word relationships in sequences more efficiently. Unlike recurrent models, attention operates on a word-to-word basis, evaluating how each word in a sequence relates to all others, including itself. By calculating the dot product between word vectors, the attention mechanism identifies the strongest relationships for each word within the entire sequence (Rothman, 2021). This is visualized in Figure 13 using the sentence, *"The animal didn't cross the street because it was too tired."* During the encoding process, self-attention allows the model to look at past and future words to improve the encoding quality. This eventually makes the model associate *"it"* with *"animal"* (Alammar, 2018). In the original Transformer model, each attention sub-layer incorporates eight parallel attention mechanisms, improving computational speed during encoding and understanding word relationships (Rothman, 2021).

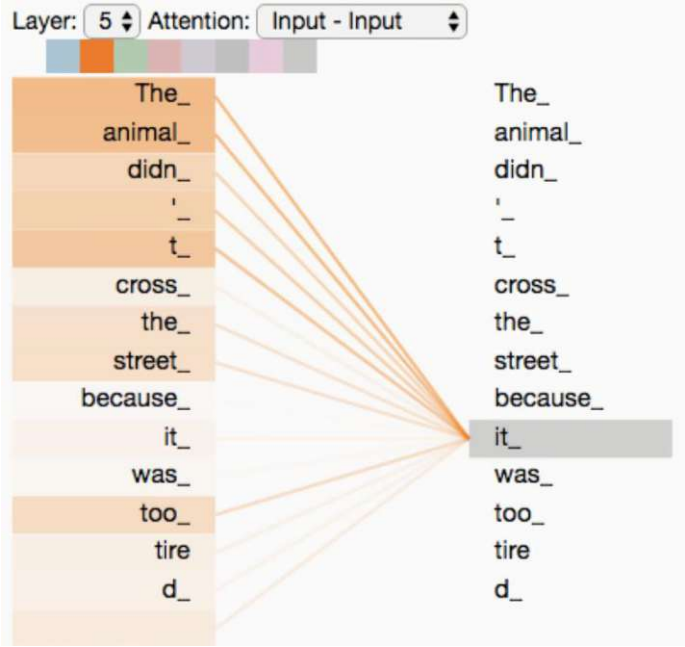


Figure 13 | The self-attention mechanism identifies the relationship between words of the input sequence during the encoding process (Jay Alammar, 2018)

Encoder stack

In the original transformer model published by Vaswani et al. (2017), the encoder and decoder consist of stacked layers, each following a consistent structure. As described in Rothman (2021), within the encoder stack, all six layers share a common architecture, comprising two sub-layers: a multi-headed attention mechanism and a fully connected feedforward network. Each of these sub-layers has a residual connection, preserving the unprocessed input as the data progresses through the layers, which is then normalized to retain essential information like positional encoding. In contrast, the embedding sub-layer exists only at the base level of the stack, ensuring stability in the encoded input throughout the subsequent layers (Rothman, 2021).

Like in other models, the input embedding sub-layer of the transformer framework transforms all input tokens into vectors of dimensionality 512 using learned embeddings (Rothman, 2021). However, as transformer networks omit recurrence and convolution, incorporating information about the word's order in a sequence becomes necessary. To address this, positional encodings in the form of sine and cosine functions are incorporated into the input embeddings at the base of both the encoder and decoder stacks. These positional encodings share the same dimension as the embeddings (512), allowing summation between both (Vaswani et al., 2017).

The multi-head attention mechanisms perform the same functions from layers 1 to 6. Compared to regular self-attention, the model can focus on multiple positions within a sequence (Alammar, 2018). The model incorporates multiple sets of attention heads. After training, each set is initialized randomly and is utilized to project input embeddings or vectors from lower encoders into different representation subspaces. This diversification improves the model's capacity to capture semantic relationships within the data (Alammar, 2018).

Finally, every attention sub-layer and feedforward sub-layer is followed by a post-layer normalization. This process comprises an add function and a subsequent layer normalization process. The add function manages the residual connections from the sub-layer's input to ensure critical information is kept through the encoding process (Rothman, 2021).

Decoder stack

The structure of the decoder layer in the transformer model mirrors that of the encoder and is the same across all six layers. Each decoder layer comprises three sub-layers: a multi-headed masked attention mechanism, another multi-headed attention mechanism connected to the encoder, and a fully connected feedforward network (Rothman, 2021). It is important to note that the decoder introduces a third key sub-layer compared to the encoder, the masked multi-head attention mechanism, which masks specific words, allowing the model to make inferences solely based on prior context without having access to subsequent parts of the sequence (Rothman, 2021). As in the encoder, residual connections connect each of the three primary sub-layers within the decoder. The embedding layer sub-layer exists solely at the stack's base level, and the output dimensionality remains constant across all sub-layers of the decoder stack (Rothman, 2021).

During the decoding process, the decoder stack generates a float vector. The transformation of this vector into a specific word is accomplished through a final linear layer followed by a softmax layer. The linear layer represents a fully connected neural network, projecting the decoder's output vector into a larger logits vector (Alammar, 2018). For instance, if the model knows 10.000 distinct English words learned from training data, the logits vector comprises 10.000 cells, including a unique word's score.

The softmax layer converts these scores into probabilities, ensuring they are all positive and sum up to 1.0. Finally, the cell with the highest probability is selected as the output for the current decoding step (Alammar, 2018).

Initially, the transformer architecture was introduced to solve sequence-to-sequence tasks like machine translation. However, today, both the encoder and decoder are used separately for solving NLP tasks, resulting in auto-encoding (encoder only), auto-regressive (decoder only), and sequence-to-sequence (encoder and decoder) models. Generally, they undergo a transfer learning process, which comprises initial pre-training on large amounts of text data to learn a general-purpose representation of the inputs and subsequent fine-tuning on a specific downstream task (Peters et al., 2019). The following sub-sections will describe the model types and the transfer learning process.

2.3.2 Sequence-to-Sequence Models

Sequence-to-sequence models, including the transformer architecture's encoder and decoder components, operate with distinct attention capabilities. Encoder attention layers can access all words in the original sentence, whereas decoder attention layers can only access words preceding a specific word in the input (Hugging Face, 2023c).

Notable sequence-to-sequence models are:

- BART (M. Lewis et al., 2019)
- Marian (Junczys-Dowmunt et al., 2018)
- T5 (Raffel et al., 2020)

BART is one of the most well-known sequence-to-sequence models. It is trained through a process involving document corruption and subsequent optimization using a reconstruction loss that leverages the cross-entropy between the output of the decoder and the original document (M. Lewis et al., 2019). Multiple transformation methods were explored, including token masking, token deletion, text infilling, sentence permutation, and document rotation. Token masking involves randomly selecting tokens and replacing them with masked elements (cf. Figure 14). Token deletion removes random tokens, requiring the model to identify the missing positions. Text infilling replaces selected text spans with masked tokens, teaching the model to predict the number of missing tokens in a span. Sentence permutation shuffles document sentences randomly, and document rotation rotates the document to begin with a randomly chosen token, training the model to recognize the document's start (M. Lewis et al., 2019).

Models using an encoder-decoder architecture and pre-trained like BART are optimal for solving tasks like sentence generation based on given inputs after fine-tuning, such as summarization and machine translation (Hugging Face, 2023c).

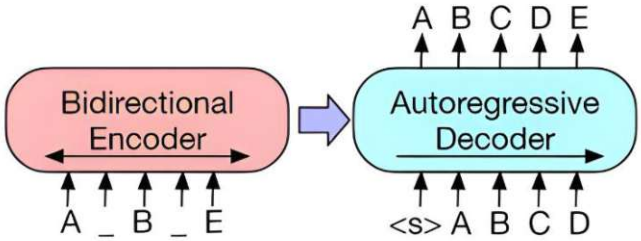


Figure 14 | The sequence-to-sequence model named BART is pre-trained on reconstructing corrupted input sequences (M. Lewis et al., 2019)

2.3.3 Auto-Encoding Models

Auto-encoding models only use the encoder of the transformer architecture combined with a new mechanism called “bi-directional” attention. It was first introduced inside the BERT (Bi-directional encoder representations from transformers) model by Devlin et al. (2019). As seen in Figure 15, it introduced a bidirectional multi-head attention sub-layer to the transformer architecture, enabling it to analyze all words in a sentence at the same time during encoding. As a result, BERT eliminates the decoder layers and incorporates masked tokens directly into the attention layers of the encoder (Rothman, 2021).

The pre-training of BERT involves two tasks, as described by Devlin et al. (2019):

- Masked language modeling
- Next sentence prediction

Masked language modeling does not require training a model with a sequence of visible words followed by a masked sequence. Using bi-directional attention, auto-encoding models can analyze a sentence with a random token mask. An example would be transforming *"The cat sat on it because it was a nice rug"* into *"The cat sat on it [MASK] it was a nice rug."* This masked token is then predicted through the multi-attention sub-layer, which processes the entire sequence (Rothman, 2021). Additionally, BERT incorporates next-sentence prediction as a pre-training method. Two new tokens, *[CLS]* and *[SEP]*, are introduced. *[CLS]* serves as a binary classification token, appended at the start of the initial sequence to predict if the second sequence follows. Positive samples are consecutive sentences from a dataset, while negative samples involve sequences from unrelated documents. *[SEP]* acts as a separation token, indicating the end of a sequence. An example, reusing the prior one and adding an additional sentence, would be *"The cat slept on the rug. It likes sleeping all day."* is transformed to *“[CLS] the cat slept on the rug [SEP] it likes sleep ##ing all day[SEP]”* (Rothman, 2021).

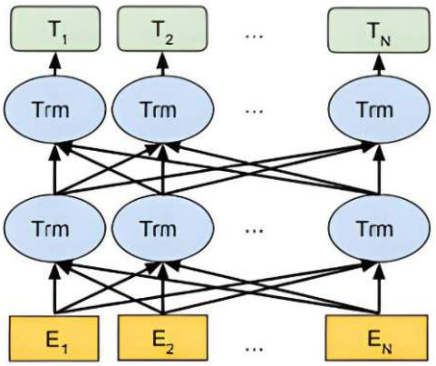


Figure 15 | Bi-direction attention enables models like BERT to access past and future words during the encoding process (Devlin et al., 2019)

After fine-tuning, auto-encoding models like BERT can be leveraged effectively in NLP downstream tasks that require a deep understanding of the whole sentence, like sentence classification, TC, and extractive QA (Hugging Face, 2023a).

2.3.4 Auto-Regressive Models

An issue of auto-encoding models is the discrepancy between the pre-training and fine-tuning process, which is caused by the absence of masked symbols in the fine-tuning data. Auto-regressive models avoid this issue by using a pre-training objective, which consists of predicting the next token conditioned on previous ones without masking (Yıldırım & Asgari-Chenaghlu, 2021).

As described by Alammar (2019), these models consist of multiple transformer decoder blocks, each featuring a masked multi-head self-attention layer and a feed-forward layer. The final decoder block's output is processed through a linear layer and a softmax function to predict the next word (Alammar, 2018). Figure 16 illustrates the architecture and the prediction process. The model's predictive process begins with the start token *[s]* and continues until the last predicted token, anticipating the next token in the sequence. For instance, during the initial round, the model utilizes *[s]* to forecast *[robot]*. Subsequently, the input sequence is extended to *{[s], [robot]}* as *[robot]* has been predicted (Bailan He, 2020).

The generative pre-trained transformer (GPT) models, designed and trained by OpenAI, are today's most prominent auto-regressive language models. OpenAI pre-trained them on unlabeled data, allowing the attention layers to learn language from unsupervised sources. This approach focused on creating a task-agnostic model, minimizing dependence on labeled data, which is costly to create, and the need for fine-tuning for specific downstream tasks (Rothman, 2021). This process happened in four phases, as outlined by Rothman (2021):

1. **Fine-tuning:** Initially, transformer models were pre-trained and subsequently fine-tuned on specific tasks.

2. **Few-shot:** The pre-trained models utilize task demonstrations for conditioning instead of weight updating through fine-tuning. This method enabled text completion based on the provided context.
3. **One-shot:** Building upon few-shot, this phase involved the model making inferences with only one demonstration of the task, with no weight updating allowed.
4. **Zero-shot:** The trained model performed downstream tasks without any task demonstrations.

This approach achieved a strong performance across a spectrum of NLP tasks and benchmarks in zero-shot, one-shot, and few-shot scenarios and nearly rivaled that of fine-tuned state-of-the-art systems in some cases (Brown et al., 2020). It also started the successful development of many other auto-regressive models like PaLM (Chowdhery et al., 2022) and Llama (Touvron et al., 2023). However, many of them are not open-source and are only accessible through APIs, limiting the possibility of research being conducted by the scientific community.

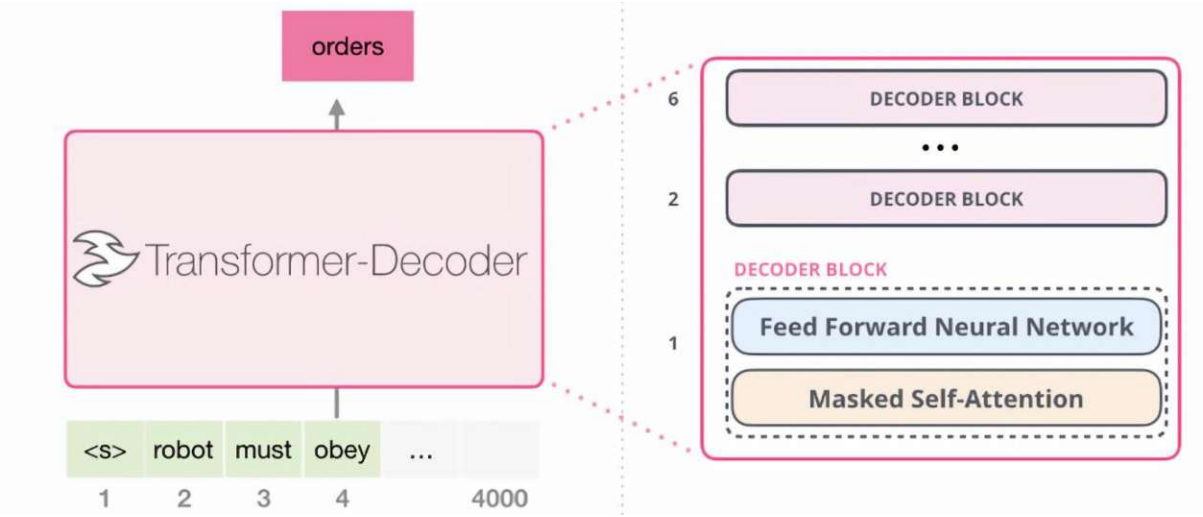


Figure 16 | The predicate process of an auto-regressive model (Jay Alammar, 2019)

2.3.5 Transfer Learning

Transfer learning harnesses existing knowledge from diverse settings, such as different tasks, languages, or domains, to assist in solving new problems. This method mirrors human learning processes where prior knowledge is leveraged in new downstream tasks. This approach is essential in democratizing technological access, allowing individuals with limited resources to achieve state-of-the-art NLP performance (Azunre, 2021).

Architecturally, it divides the model into a body and a task-specific head, where the body learns general features from the source domain. These learned weights initialize

a new model for the new task, as seen in Figure 17. Compared to traditional supervised learning, this method yields high-quality models that are efficiently adaptable to various downstream tasks with significantly reduced labeled data requirements (Tunstall et al., 2022). Thus, the universal language model fine-tuning technique (ULMFiT) was introduced to adapt neural-network-based language models to new tasks. Initially designed for TC, ULMFiT allows the fine-tuning of language models for any target applications and consists of three main steps (Azunre, 2021; Tunstall et al., 2022):

- **Pre-training**: The language model learns to predict the next word based on preceding words using unlabeled text sources (auto-regressive models) to denoise a corrupted text sequence or to predict the next sentence (auto-encoding models). This process provides the model with general language knowledge.
- **Domain adaptation**: The model is adapted to the in-domain language corpus, maintaining the pre-training objective in the target corpus.
- **Fine-tuning**: The language model is refined with a classification layer tailored to the target downstream task.

These steps made transfer learning effective for NLP applications. In the case of BERT models, the cost disparity between pre-training various model sizes from scratch is significant, with the largest model training costing millions of dollars. However, the advancement in transfer learning techniques empowers individuals to reuse this existing knowledge in personal projects, requiring only a few hours or a minimal financial investment for fine-tuning (Azunre, 2021).

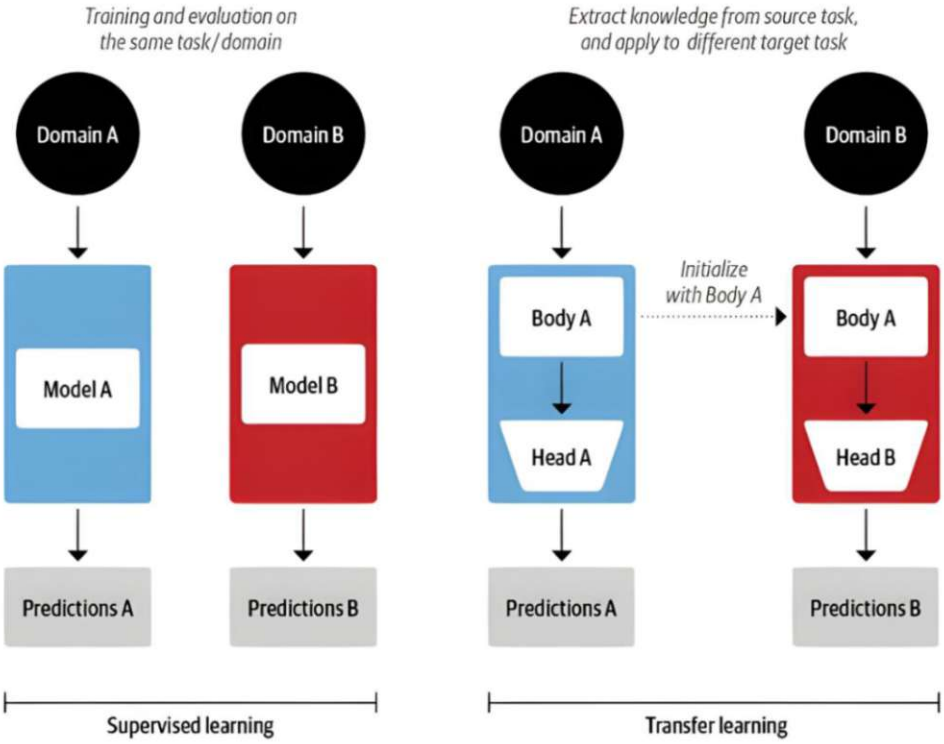


Figure 17 | The concept of transfer learning in the field of NLP (Tunstall et al., 2022)

3 State-of-the-art Domain-specific Fine-tuning of Pre-trained LLMs

As outlined in Section 2.2.3, chatbot systems comprise several interconnected components. These architectural components must perform a range of NLP tasks to enable the chatbot to communicate effectively with its users. Tasks that are commonly performed by the NLU, dialog management, and NLG component (cf. Figure 11) are:

- **NLU component:**
  - Intent detection (Text classification)
  - Named entity recognition
  - Extractive QA
- **Dialog management component:**
  - Information retrieval
- **NLG component:**
  - Text generation

As stated in the previous section, transformer-based pre-trained LLMs have triggered a paradigm shift within the field of NLP. They yield state-of-the-art performance in downstream NLP tasks used in chatbot frameworks after being fine-tuned. Consequently, integrating these pre-trained and fine-tuned LLMs into such a framework is a non-trivial undertaking. This task is further compounded by the rapidly evolving landscape of NLP research and the ongoing release of different pre-trained LLMs. Coupled with the exponential growth of scientific literature, this dynamic landscape causes a challenge in selecting the most suitable pre-trained model and downstream task for fine-tuning.

Consequently, this section explores state-of-the-art domain-specific fine-tuning strategies of pre-trained LLMs within the context of chatbot systems through a systematic literature review. The resulting comparison and analysis of fine-tuning processes and chatbot integration approaches in existing publications provide a supporting tool in the decision-making process of early chatbot development to leverage existing work in the field of NLP effectively.

3.1 Methodology of the Systematic Literature Review

The applied methodology for the literature review is based on the works of Peres et al. (2020) and Zonta et al. (2020), which comprise reviews about AI and predictive maintenance in Industry 4.0. Both works summarize the application of rapidly evolving technologies in real-world manufacturing environments, identifying their main methods and strategies. The gap in research is identified as a first step for carrying out the literature review, and research questions are created. Analog to (Peres et al., 2020),

this work follows the widely used reporting framework described in the PRISMA statement (Page et al., 2020) because of its simplicity and transparency. Its general structure for identifying studies via databases and registers is depicted in Figure 18.

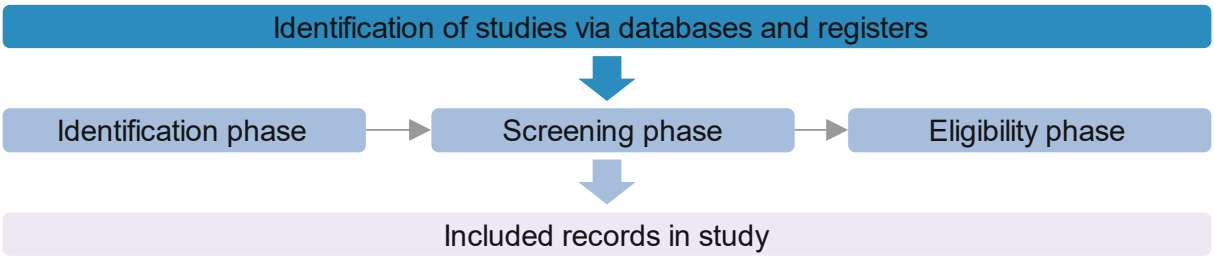


Figure 18 | Identification of studies via databases and registers following the PRISMA statement

The search string was constructed through an iterative procedure by applying it to the Google Scholar database and evaluating the top search results until a good trade-off between the number and relevancy of publications was reached. The resulting string was applied and adapted to the Google Scholar, Scopus, and IEEE database on 28.04.2023 and is shown in Figure 19.

(“chatbot” OR “virtual assistant” OR “conversational agent” OR “conversational AI”) AND (“large language model” OR “language model”) AND (“maintenance” OR “industry” OR “manufacturing”) AND ("transfer learning" OR "fine tuning")

Figure 19 | Search string for identifying relevant literature about domain-specific fine-tuning of LLMs and their integration into chatbot frameworks

Citations and patents are excluded to retrieve only relevant research papers for this review. Further, only papers that were published after the release of the transformer-based model BERT are included. It achieved widespread recognition for outperforming previous state-of-the-art language models and provided the foundation for many future transformer-based models (Bommasani et al., 2021).

Afterward, the 100 most relevant publications of each database that fulfilled the mentioned criteria were retrieved from the databases. The resulting records have been exported, aggregated, and duplicates inside each database and between the databases have been removed. Following the identification phase, the screening phase has been carried out by the author. The screening criteria listed in Table 2 are applied to each record's title, abstract, and keywords. SC1 ensures that the publication addresses the fine-tuning of LLMs in general, independent of the use of domain-specific data. SC2 excludes works that do not provide a unique fine-tuning solution for LLMs.

Table 2 | Screening criteria were applied to the title, abstract, and keywords

ID	Screening criteria
SC1	The abstract or title must address fine -tuning of LLMs
SC2	Records must not be solely a survey, review, or roadmap

The records that met the criteria SC1-2 went through the eligibility process. Here, the eligibility criteria listed in Table 3 were applied to the title, abstract, and keywords and to the full text of each record. EC2 ensures that fine-tuning was performed on closed domain data and that the record is relevant to the work's practical part – developing a chatbot prototype. Additionally, EC3 removes all publications that show insufficient report quality regarding the pre-trained LLM, fine-tuning process, dataset, and implementation into a chatbot framework. Finally, the remaining records are included in the literature study and are thoroughly analyzed.

Table 3 | Eligibility criteria were applied to the full text of the record

ID	Eligibility criteria
EC1	Full text must be available
EC2	Addresses the integration of a fine -tuned LLM on closed domain data in a question answering system/chatbot/virtual agent
EC3	Insufficient report quality

3.2 Identification of Models and Downstream Tasks for Domain-specific Fine-tuning

After applying the search string to each database and retrieving the 100 most relevant publications, 275 records were identified for the screening and eligibility process. Eventually, 21 publications were included in the following literature study about state-of-the-art domain-specific fine-tuning strategies of LLMs in the context of chatbot systems. The complete PRISMA report shown in Figure A. 1 in the appendix further details the identification, screening, and eligibility process. A full-text analysis was carried out for each record, extracting relevant information for choosing the right model for a domain-specific chatbot integration. For general information, this includes the model itself, its source code's availability, and the chatbot's application domain. Regarding the fine-tuning strategy, information about the NLP downstream task, training data structure, and data set size is collected.

Table 4 provides a summary of the analysis, enabling various observations. Notably, each fine-tuned LLM utilized in a chatbot framework is individually listed when multiple models are employed. Fine-tuning necessitates access to the model's source code for weight modifications. Only Mehboob et al. (2023) use GPT-3 by OpenAI, which can only be accessed and fine-tuned through an API and is, therefore, considered not publicly available within this work.

Table 4 | Records included in the systematic literature review with information about the model, availability, application domain, NLP downstream task, data structure and size used for fine-tuning process to achieve domain adaptation

Author	Model	Public	Domain	Task	Data structure	Data size
Alloatti et al. (2019)	Bert	✓	Financial & Legal	Intent Classification	Question & Answers	2,300
C. Li et al. (2022)	Bert	✓	Manufacturing	Intent Classification	Labeled Questions	
Chakraborty et al. (2023)	Bert	✓	Automotive	Intent Classification	Labeled Questions	299,672
Firsanova (2021)	Bert	✓	Education	Question answering	Question & Answers	500
Firsanova (2021)	GPT-2	✓	Education	Question answering	Question & Answers	500
Kapoor & Shetty (2023)	Distil GPT-2	✓	Health	Text Generation	Question & Answers	50,000
Krishnan et al. (2020)	Bert	✓	Software Engineering	Concept Recognition	Labeled Sentences	3.700
L. Wang et al. (2021)	GPT-2	✓	Health	Text Generation	Dialogs	306
Lin et al. (2022)	Bert	✓	Construction	Question answering	Question & Answers	10,000
Mehboob Knightec et al. (2023)	GPT-3	×	Health	Text Generation	Images + Sentences	2,900
Minutolo et al. (2022)	Bert	✓	Health	Intent Classification	Labeled Sentences	50
Montenegro & da Costa (2022)	Sentence Bert	✓	Health	Semantic Text Similarity	Sentence Pairs	1,500
N. Wang et al. (2022)	RoBERTa	✓	Construction	Intent Classification	Labeled Queries	2,065
Obadinma et al. (2023)	Bert	✓	Financial & Legal	Intent Classification	Labeled Queries	13,000
Pan et al. (2021)	RCI	✓	Aviation	Question answering	Questions & Tables	9,092
Tang et al. (2019)	Bert	✓	Customer Service	Intent Classification	Labeled Questions	25,000
Vegupatti et al. (2020)	Bert	✓	Biomedical	Question answering	Question & Answers	600
Vishwanathan et al. (2023)	ConvBERT	✓	Customer Service	Semantic Text Similarity	Question Pairs	16,110
Vishwanathan et al. (2023)	Sentence Bert	✓	Customer Service	Semantic Text Similarity	Question Pairs	16,110
Wu et al. (2019)	Multi-lang Bert	✓	Customer Service	Intent Classification	Labeled Sentences	3,500
Y. Li et al. (2023)	Llama	✓	Health	Text Generation	Dialogs	10,0000
Yan & Nakashole (2021)	GPT-2	✓	Health	Text Generation	Question & Answers	260,000
Yuan & Afli (2021)	Bert	✓	Health	Sentiment Analysis	Labeled Sentences	58,000

Figure 20 (a) illustrates a trend in domain-specific fine-tuning of LLMs, with predominant research occurring in the health and customer service sectors. The label customer service domain includes all records, which did not further specify its industry. This highlights the imbalance in current LLM research and the need to create more domain-specific models such that a wide range of industries can leverage state-of-the-art NLP technologies for their business needs. The unbalanced distribution of research focus can be attributed partially to the availability of publicly accessible datasets for specific industries. A lot of conversational data exists in customer service (Feigenblat et al., 2021) and in the health domain (Zeng et al., 2020) from recording either customer requests and answers or medical dialogue. For other domains in which information is conveyed more structured, like in the maintenance industry through maintenance logs, such data generally does not exist.

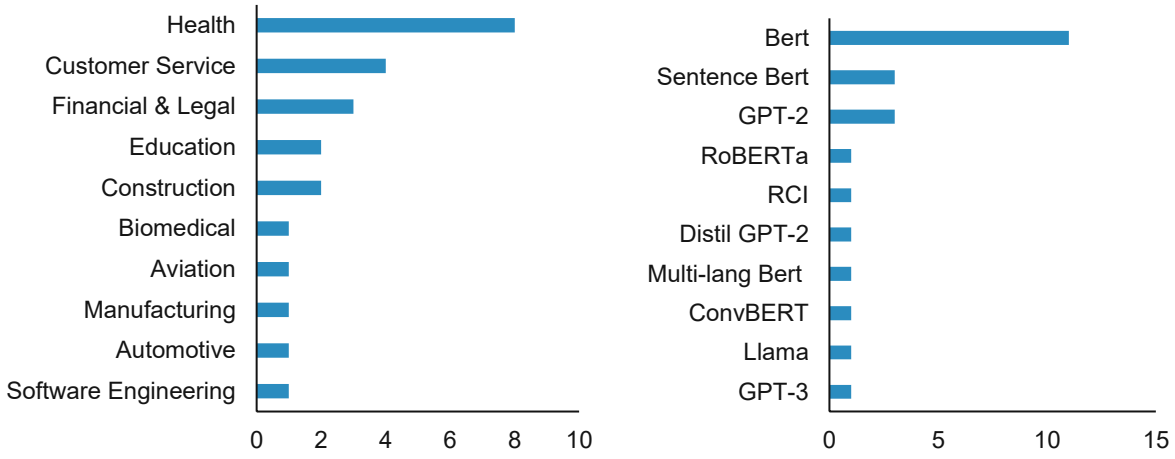


Figure 20 | (a) Application domain of fine-tuned LLMs (b) Models fine-tuned on domain-specific data

Table 4 shows that, depending on the model and downstream task, up to 299,672 labeled data points are required to achieve state-of-the-art fine-tuning results. Large datasets for NLP projects are expensive to create, requiring multiple domain experts to review, annotate, and label data (Hendrycks et al., 2021). Such resources are provided mainly by large corporations such as Google, Amazon, and Facebook or academic institutions. However, they are not available to smaller-sized companies. Data augmentation is a possible solution to this scarcity problem, which will be discussed in more detail in Section 4.2.3.

Like the application domain, the distribution of fine-tuned LLMs in the context of industrial chatbots has a low variance, as shown in Figure 20 (b). 16 out of 21 publications use the pre-trained language model BERT or a BERT-based model for at least one or more chatbot modules, while the rest use GPT-based models. This is a surprising observation regarding the exponential rise of decoder-only models in the last couple of years. This finding is further underscored when comparing all identified models with the extensive model summarization from (Rastogi, 2023). Only 6 of the 40

listed LLMs that were released until May 2023, when the review was conducted, are fine-tuned with domain-specific data. This indicates that only a few of all currently released models are suited for domain-specific fine-tuning.

Since the fine-tuning process is a computationally resource-heavy task, its model and downstream task should be chosen early in chatbot development. The main downstream tasks that were identified for domain-specific fine-tuning are:

- Named entity recognition (NER)
- Question-answering (QA)
- Semantic text similarity (STS)
- Text generation (TG)
- Text classification (TC)

By further analyzing Table 4, the average distribution of use of each model for a specific downstream task is summarized in Table 5. For the sake of readability, the models that appeared more than once are listed individually, while all other models are grouped. BERT models provide a wide range of applications for various downstream tasks. More than half of the time, it is used for TC tasks, followed by QA tasks. NER tasks are less common. However, they are not prevalent with any other model. The GPT-2 model is only used for generative tasks like TG and QA. In contrast, SentenceBERT models find utility in STS tasks due to their underlying architecture. GPT models, due to their auto-regressive nature, excel in TG tasks but exhibit comparatively lower performance in text understanding tasks compared to auto-encoding models like BERT (Lipenkova, 2022). Therefore, it might be advantageous to use auto-encoding models for domain-specific fine-tuning. These models can be fine-tuned for a broad array of downstream tasks, including extractive QA, sentence classification, or word recognition, ensuring comprehensive in-domain NLU.

Table 5 | Amount of fine-tuned LLMs for each identified downstream task relative to their appearance

Model/Task	NER	QA	STS	TG	TC
Bert	0.09	0.27	0.00	0.00	0.64
Sentence Bert	0.00	0.00	1.00	0.00	0.00
GPT-2	0.00	0.33	0.00	0.67	0.00
Others	0.00	0.14	0.14	0.43	0.29

Table 6 provides a dataset size reference required for domain adaptation through fine-tuning, indicating a minimum requirement of 500 rows of data independent of the downstream task. Among the examined records, QA requires the smallest dataset, while STS and TG demand larger ones. This range is essential to consider during the developmental phase, particularly for industries with limited data resources.

Table 6 | Average size of dataset used for domain-specific fine-tuning for each downstream task

Model/Task	NER	QA	STS	TG	TC
Bert	2150	5250	-	-	66337
Sentence Bert	-	-	95870	-	-
GPT-2	-	500	-	130153	-
Others	-	9092	16110	50967	2782.5

Finally, before starting a fine-tuning process, it must be clear how the LLM will be implemented into a chatbot framework and what the benefits to existing methods are. Consequently, the following subsections conduct a detailed analysis of selected publications for each identified downstream task. It clarifies their fine-tuning methodologies and the subsequent incorporation of the final LLM into a chatbot framework.

3.2.1 Text Generation

Fine-tuned LLMs for TG tasks are currently implemented as an end-to-end chatbot solution and a TG module only. The critical difference is that an end-to-end solution is fine-tuned on domain-specific dialogs and only receives the user's utterance directly as input. In contrast, a text-generating module receives a prior prompt from the chatbot system, which can include additional information retrieved from a knowledge database. For domain-specific chatbots, this approach seems superior. However, additional NLU modules are generally required and increase overall system complexity.

Kapoor & Shetty (2023); L. Wang et al., (2021); Yan & Nakashole, (2021) use the pre-trained GPT-2 model, fine-tuned on medical dialogs, as an end-to-end chatbot solution. Even though this method can generate human-like responses in the medical field, it is unable to retrieve information from a knowledge base. Y. Li et al. (2023) used the latter method to build a medical question-answering chatbot, using pre-trained LLaMA in combination with an external knowledge base. Keywords are extracted by an NLU system and are matched against the medical knowledge base. The top result is inserted into a pre-defined prompt, which generates the answer to the user's question. This mitigates the phenomenon of hallucinations for unfamiliar inputs and controls the informational content included in the response while increasing system complexity. To better understand the user's utterance and generate more accurate answers, the LLM was additionally fine-tuned on 100,000 doctor-patient conversations.

3.2.2 Text Classification

In the context of LLM-based chatbot design, TC mostly means identifying the type of question posed by the user and calling for a specific action or response. Unlike TG tasks, a separate chatbot module is required for response generation.

Minutolo et al. (2022) developed a conversational agent for querying patient information leaflets using a retrieval-based architecture. A fine-tuned BERT model classifies the user's intent of a question, and its tag is used to trigger a specific action by the dialog management system and query for specific information in the domain knowledge repository. It was fine-tuned on labeled user inputs with a dataset size of 50 for each possible intent. Eventually, a response is generated using predefined templates and the previously retrieved information. Obadinma et al. (2023) extend the approach of a retrieval-based architecture by replacing both the information retrieval and response generation module with an LLM itself. While the intent classification module remains a fine-tuned BERT model, trained on a finance dataset with 13,000 labeled queries, the retriever uses SentenceBERT to find relevant passages, including the user's answer from the knowledge base. After reranking the results, a GPT-2 model is fed the input query, the intent, and the relevant context to perform the final response generation task, providing a more user-oriented response generation. Compared to (Minutolo et al., 2022), this system architecture is less complex. However, it still falls short compared to approaches using only LLMs fine-tuned for domain-specific TG tasks.

3.2.3 Semantic Text Similarity

STS tasks evaluate the similarity between two texts. Using BERT's cross-encoder, such a task would be very time-consuming and unviable if the pool of sentences for comparison is too large. Therefore, Reimers & Gurevych (2019) developed SentenceBERT, a modification of the BERT architecture using Siamese networks shown in Figure 21. It can be used to compute the cosine similarity between two sentences efficiently and thus can be applied for information retrieval in chatbot systems.

Like Obadinma et al. (2023) and Montenegro & da Costa (2022), Vishwanathan et al. (2023) used a SentenceBERT model for information retrieval for question-answering tasks but further fine-tuned it on pregnancy and customer service data to achieve domain adaption. Vishwanathan et al. (2023) found that fine-tuned bi-encoders work best for domain-specific FAQ retrieval tasks compared to regular pre-trained models across multiple tenant datasets. Their fine-tuning strategy involved only changing the weights of the final layer for each tenant dataset and sharing the base model with all others. The datasets are comprised of question pairs, including a label indicating if both sentences belong to the same class. This results in a FAQ retrieval system, which is superior to TC systems to link each question to its respective answer by intent. To improve response precision, which is critical in the health domain, Montenegro & da Costa (2022) used an information retrieval algorithm for searching an ontology. The ontology is based on medical records and stores data as an answer to the corresponding question. It returns a set of documents based on the extracted entities

from the user question using the information retrieval algorithm. The SentenceBERT's task is to re-rank the answers and select the correct one. Fine-tuning for a better understanding of pregnancy-related texts was done using 600 positive and 900 negative sentence pairs. The pairs were created by a data augmentation strategy, which took random sentences or passages from a domain-specific document and produced similar semantic pairs through a SentenceBERT model. Compared to fine-tuned models without a data augmentation strategy, their performance in information retrieval of domain-specific data increased significantly.

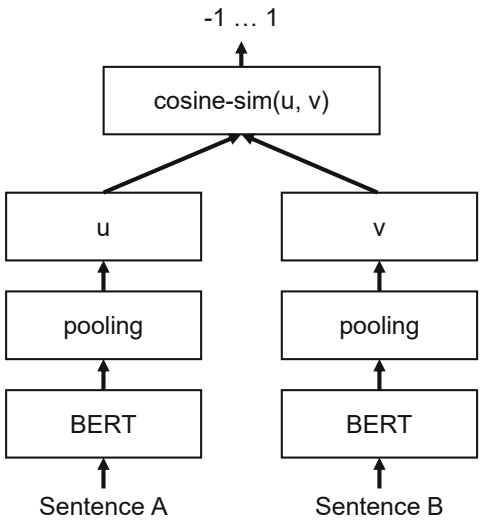


Figure 21 | SentenceBert leverages a Siamese architecture for efficient computation of text similarity measures

3.2.4 Named Entity Recognition

As seen in Table 4, fine-tuning pre-trained LLMs for domain-specific NER tasks appeared less often in the reviewed literature. Only Krishnan et al. (2020) developed a concept extractor for a system engineering virtual assistant using a fine-tuned BERT model. The concept recognition task is identical to an NER task. However, it is trained to extract domain-specific entities instead of common physical entities, such as names of persons, locations, or organizations. It is combined with a relation extraction algorithm between concepts to enable automated knowledge graph creation. Fine-tuning of the BERT model was performed using 3,700 annotated sentences on a word-token level.

3.2.5 Question Answering

There are different QA approaches in NLP that must be considered individually when choosing a downstream task for domain-specific fine-tuning:

- Extractive QA
- Open generative QA

- Closed generative QA.

While extractive QA models just extract the answer to a given user input from the provided context (e.g., a text passage), generative QA models generate the answer based on the user's input and context (Open generative QA) or just the user's input (Closed generative QA). BERT-like models mostly solve extractive QA tasks, and decoder-only models like GPT-2 are used for generative QA tasks. The fact that an additional context, which should contain the answer to the user's questions, is provided to the model makes this task less prone to hallucinations and, therefore, better suited for systems that require high answer accuracy.

Lin et al. (2022) used a fine-tuned BERT model to build a question-answer system for building information modeling. It receives ranked paragraphs from an inverted search algorithm as context and subsequently extracts the answer. The authors used 3,334 text paragraphs that comprised 10,002 questions for the fine-tuning task. Additionally, data augmentation was applied to increase the diversity of certain data types, reduce overfitting, and improve the generalization ability. Firsanova (2021) compared the performance of a fine-tuned BERT and GPT-2 model for question-answering about Asperger syndrome. The dataset for training comprised 500 question-and-answer pairs. To train the model to ignore certain question types, 10% were tagged as unanswerable, using a Boolean operator for the BERT model and a specific answer for the GPT-2 model. The author found that BERT retrieves accurate answers. However, large datasets for fine-tuning and additional chatbot modules for answer creation are required. On the other hand, the GPT-2 model required fewer data, but the generation of false answers is more prevalent compared to the BERT-based solution. Since documents for context retrieval do not always contain text passages but can be comprised of tables as well, Pan et al. (2021) present an end-to-end transformer-based table QA system. It is an adjusted BERT-based model that is fine-tuned for different benchmarks in combination with an inexpensive table retrieval algorithm. Generally, table retrieval and QA over tables are studied separately, but using this method, the authors are to directly answer questions based on a table database by identifying relevant cells.

Overall, the downstream task of extractive and open generative QA enables LLM-based chatbot architectures to act as knowledge-sharing tools without storing information in the weights of the model itself. This is a distinct advantage compared to other tasks because it avoids the issue of expensive re-training of the model each time new knowledge is added.

3.3 Summary

In this chapter, a systematic literature review has been conducted to identify state-of-the-art domain-specific fine-tuning methods. In contrast to existing works that rank

different LLMs concerning their task-specific performance (Lipenkova, 2022), this thesis compares pre-trained LLMs regarding their potential to be fine-tuned on domain-specific data and how they are integrated into chatbot frameworks.

Twenty-one relevant scientific publications have been systematically analyzed to extract information about their domain-specific fine-tuning approaches. The findings revealed a predominant use of open-source models such as BERT, GPT-2, and SentenceBert after fine-tuning for specific downstream tasks, primarily due to their accessibility. Notably, the required dataset sizes varied significantly between tasks and models. In response to these observations, this thesis formulated a guideline, presented in Figure 22, outlining the most suitable open-source pre-trained LLMs for domain-specific fine-tuning.

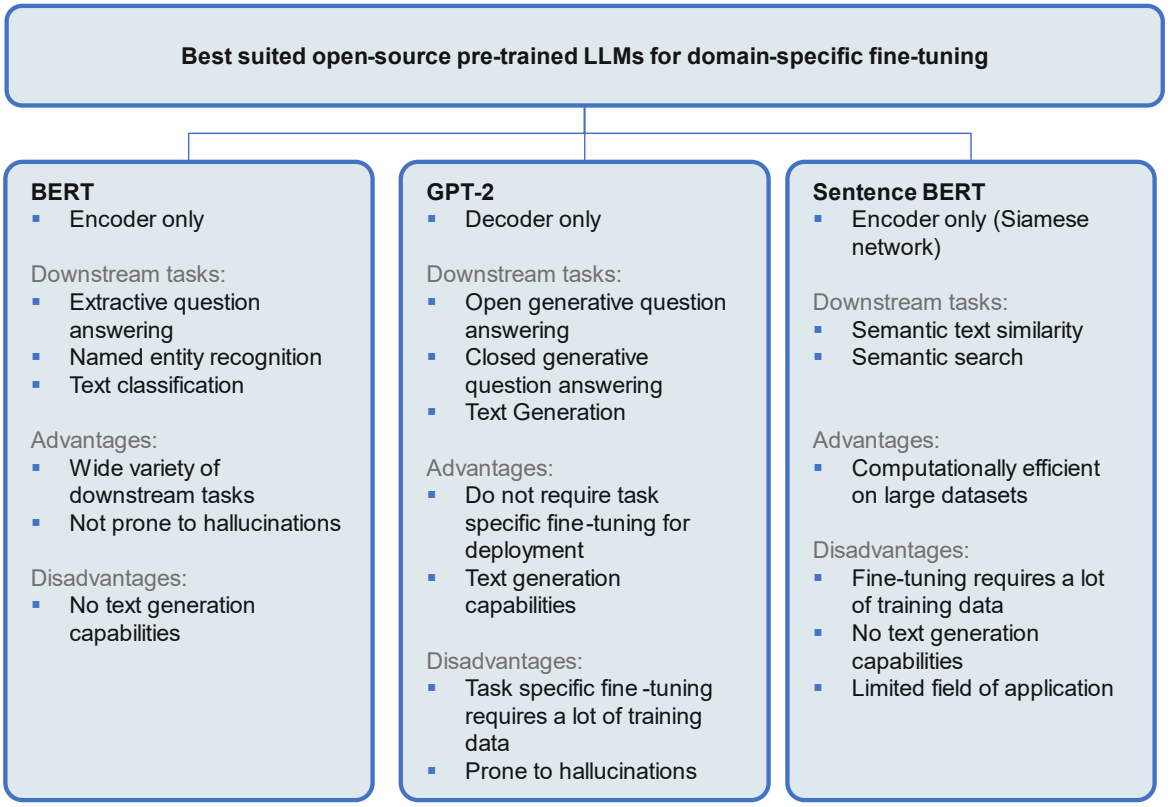


Figure 22 | Identification of the best-suited open-source pre-trained LLMs for domain-specific fine-tuning

The selected models, BERT, GPT-2, and SentenceBert, cover many potential downstream tasks for fine-tuning. The advantages of BERT models emerge from their encoder-only architecture, providing them with a deep textual understanding. Furthermore, BERT models demand relatively minimal data for fine-tuning compared to GPT-2 and SentenceBERT models. Their auto-encoding nature also mitigates the risk of hallucinations, improving prediction accuracy without any TG capabilities. In contrast, GPT-2 excels particularly in TG tasks, notably in open generative QA. However, it is important to note that GPT-2's prediction accuracy for specific tasks might be influenced by its decoder-only architecture. Nonetheless, this characteristic makes such models task-agnostic, removing the need for fine-tuning for deployment.

However, a large volume of data is typically required in scenarios where fine-tuning is necessary for domain adaptation. Finally, SentenceBERT models achieve unrivaled performance in information retrieval tasks, specifically those involving semantic search or text similarity computation, compared to BERT and GPT-2. However, their application domain is restricted, and fine-tuning generally requires a large dataset.

In the subsequent chapter, this holistic view will form the foundation for the decision-making process in the development of an LLM-based industrial maintenance chatbot. It facilitates the comparison of the chatbot's domain-specific requirements with the attributes and fine-tuning capabilities of each open-source model.

4 Development of a Maintenance Chatbot

This thesis aims to develop an LLM-driven industrial maintenance chatbot that supports users during the planning and execution of maintenance operations by answering maintenance-specific questions. In this context, three general requirements for the industrial maintenance chatbots have been identified:

1. It should generate answers to maintenance-specific questions such that it provides effective support by providing distinctive information in the field of industrial maintenance.
2. It should adapt to the specific terminology and question structure prevalent in the field of industrial maintenance through fine-tuning on a limited maintenance corpus.
3. The generated answers should be as accurate as possible, providing reliable and trustworthy information to the user.

Comparing the first requirement to the results of the preceding section, which analyzed state-of-the-art domain-specific fine-tuning strategies, question-answering has been identified as the most fitting downstream task for fine-tuning on industrial maintenance data. Regarding the choice for the underlying LLM, GPT-2 and BERT are best suited when looking at Figure 22**,** matching the first and second requirements. However, as described in Section 3.2.5**,** there are different variations in how the answer to a question is generated. Considering the third requirement, that the generated answers should be as accurate as possible, the model that performs extractive QA should be favored over ones that perform generative QA. Consequently, a BERT-like architecture is implemented as the core LLM of the developed chatbot in this thesis. This underscores the importance of prioritizing answer accuracy and mitigating the potential for hallucinations, aligning with the general chatbot requirements stated above and the overarching research objectives.

Following the fine-tuning process, the LLM will be implemented into a Rasa framework, as depicted in Figure 23, which is the most popular open-source chatbot framework and widely utilized in the domain of chatbot development for constructing AI-driven conversational agents. It includes two primary components: an NLU component that processes the user's input and a dialog management component that chooses the action based on the user's past inputs and training data (Rasa Technologies, 2023b). Within this framework, the fine-tuned BERT model will be employed as a specific action that provides maintenance-specific question-answering capabilities and can be called by the dialog management system.

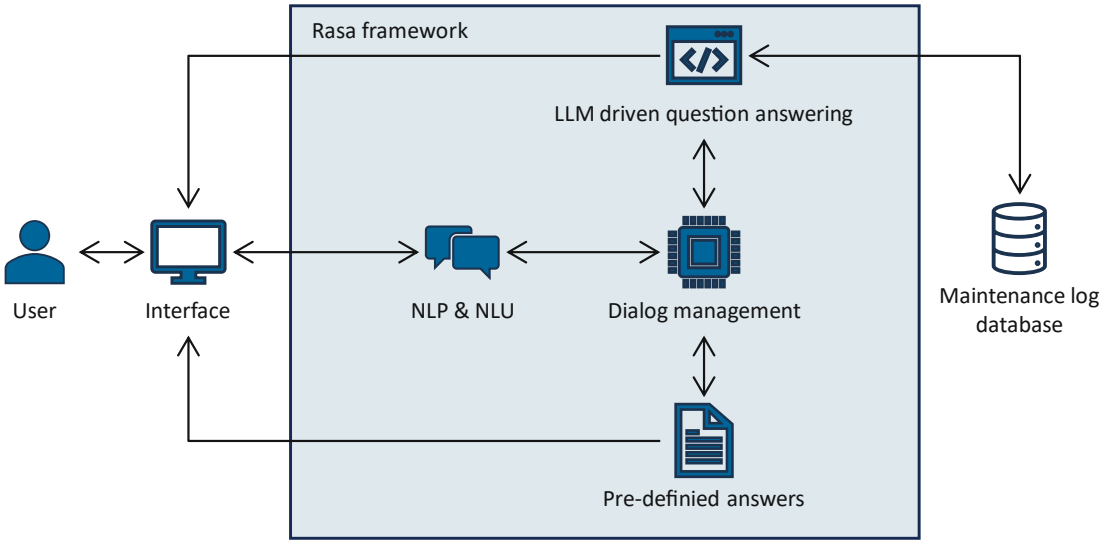


Figure 23 | The general chatbot architecture for the industrial maintenance chatbot

The following sub-section of this chapter introduces the architecture of the BERT-based TaPas model (cf. Section 4.1), which is optimized for extractive question-answering tasks on structured tabular data. In Section 4.2, the maintenance data and crafting of the training data set used for fine-tuning the TaPas model are described in detail. Additionally, increasing the dataset using data augmentation methods and its impact on model accuracy is discussed. Finally, the fine-tuning method and the Rasa chatbot architecture, including implementing the TaPas model, are described.

4.1 The TaPas Model

TaPas, which stands for “table parsing,” is an extension to BERT’s model architecture that can perform question-answering tasks over tables. Generally, this is considered semantic parsing of questions, which maps natural language questions to database queries (Yih et al., 2014). In the case of table parsing, it describes the conversion of natural language utterances to logical forms that are executed against the table to retrieve the correct answer. Until now, such semantic parsers were trained on large, annotated datasets that pair natural language questions with logical forms, which are generally very expensive to create. Instead, the TaPas model omits the necessity of creating logical forms by learning tabular query operations from natural language directly (Herzig et al., 2020).

To achieve this, two classification layers are added to BERT’s base architecture, which are used for selecting cell and aggregation operators. Tables are flattened and split into tokens before they are encoded, using positional embeddings, and concatenated after the tokenized query (Herzig et al., 2020). Figure 24 shows the extended model as well as the cell selection and aggregation operator prediction. The question and cell tokens are denoted as T_i and $Cell_i$, respectively, and marked red. [CLS] is a special classification token and [SEP] is a separation token between the question and table. The learned embeddings of the input tokens are denoted with E_i and marked yellow.

The final hidden layer of each input token is marked green and is responsible for the output (Devlin et al., 2019). The model's output is a probability score P for each cell that it will be part of the final answer (top right of Figure 24) and a score indicating which aggregation operator will be applied (top left of Figure 24) (Müller, 2020). To give a final answer to the query, the model selects the table cells with a probability larger than 0.5 and applies the predicted aggregation operator to them (Herzig et al., 2020).

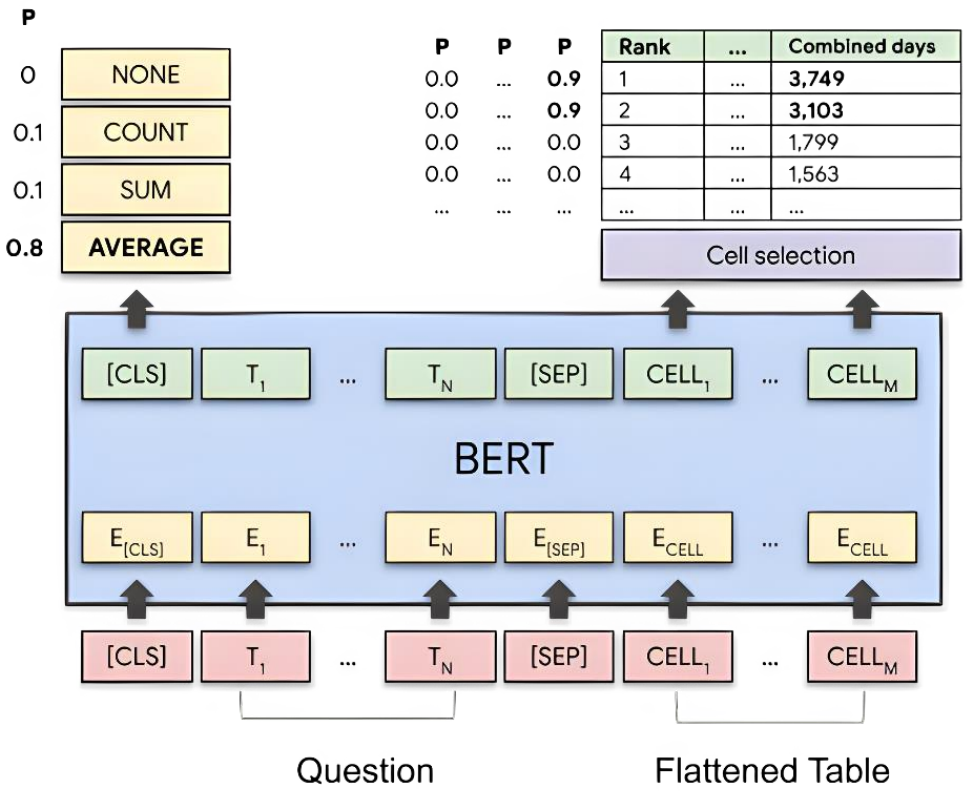


Figure 24 | TaPas architecture (Müller, 2020)

The advantage of this architecture compared to other methods for reasoning over tables is its simplicity because only an encoder with no auto-regressive decoder is used, it benefits from pre-training, and it can perform aggregation and handle a conversational setup (Herzig et al., 2020).

TaPas is pre-trained on 6.2 million text-table pairs sourced from Wikipedia, which allows it to learn correlations between the text and its respective table, cells, and header. The pre-training objective is identical to the BERT model, which is to predict masked tokens from the text and the table based on the textual and tabular context (Herzig et al., 2020).

Herzig et al. (2020) performed fine-tuning of the pre-trained model for semantic parsing in on the following table reasoning datasets:

- **WIKITQ:** This dataset includes 22,033 complex questions on Wikipedia tables, which require reasoning and data operations such as aggregation, comparison, and arithmetic computation (Pasupat & Liang, 2015).

- **SQA:** This dataset includes 17,553 question-answer pairs on Wikipedia tables in a conversational setup that contain references to previous questions or answers (Iyyer et al., 2016).
- **WIKISQL:** This dataset includes 80,654 hand-annotated questions and SQL queries across 24,241 tables from Wikipedia (Zhong et al., 2017).

The results show that fine-tuned TaPas models achieve better or comparable results to state-of-the-art semantic parsers. Despite these impressive results, the model still limits the table size, which can be encoded. It can only handle a single encoded table that fits into the model's memory (Herzig et al., 2020). Therefore, other solutions are required to handle large tables and databases, which are addressed in the later sections of this work.

4.2 Maintenance Data

Comprehensive hand-annotated datasets are utilized when developing and evaluating novel LLMs. Two very popular datasets are:

- **SQuAD 2.0:** A question-answering dataset that combines 100,000 answerable and 50,000 unanswerable questions to evaluate extractive reading comprehension systems. To achieve a high score, systems not only have to answer questions correctly but also determine when no answer is supported by the provided paragraph (Rajpurkar et al., 2018).
- **GLUE:** This is a general language understanding dataset to evaluate models across a wide range of NLU tasks. It includes tasks such as sentiment analysis and semantic similarity. It favors models that are trained jointly on the included tasks, sharing general linguistic knowledge between them (A. Wang et al., 2019).

As NLP technologies continue to progress and integrate into diverse industries, an expanding range of domain-specific datasets is becoming publicly accessible. However, in specific domains and for specific tasks such as industrial maintenance and table question-answering, publicly available annotated datasets are not available to the best of the author's knowledge.

Given the cost-intensive nature of creating extensive hand-annotated question-answering datasets for the purpose of fine-tuning, the significance of synthetic dataset generation techniques is emphasized by Rajpurkar et al. (2016). Gholami & Omar (2023) proposed a template-based question generation approach that effectively improves model training in scenarios where real-world annotated data is sparse. Given these findings, a rule-based dataset creation toolkit has been developed for this thesis, reducing the overall costs required for fine-tuning LLMs. It handles tabular maintenance data to create a semi-synthetic maintenance table question answering

dataset for fine-tuning LLMs such as TaPas. Its overall structure and functionality are outlined in Figure 25.

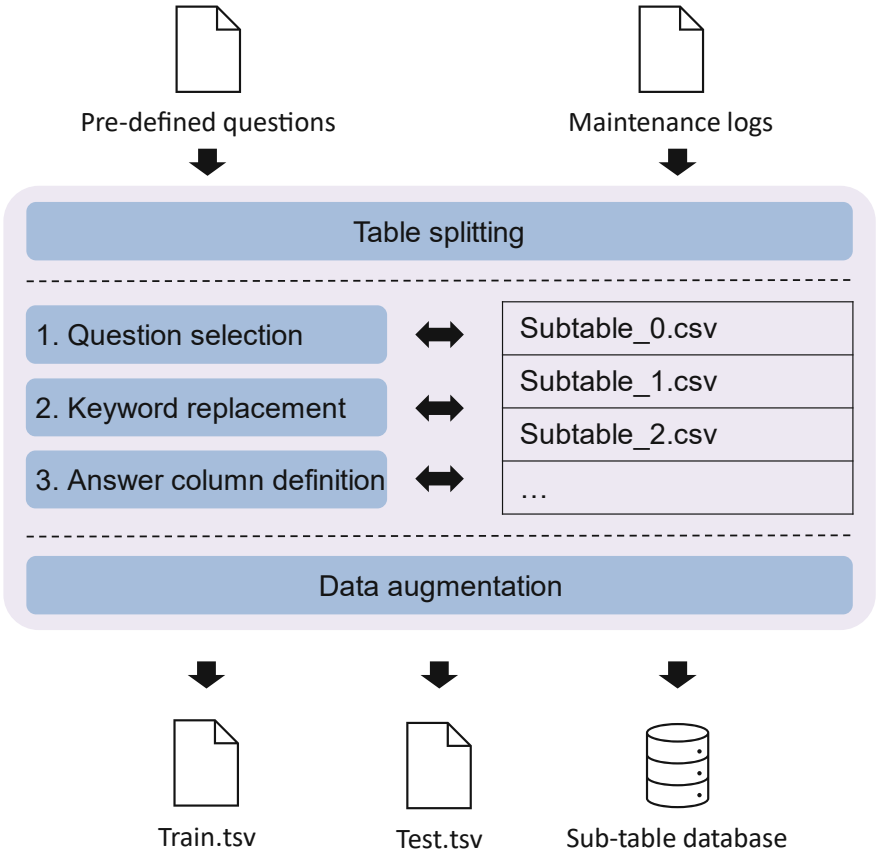


Figure 25 | Rule-based method for crafting table question-answering datasets

Firstly, this toolkit facilitates splitting the input dataset into manageable sub-tables compatible with the TaPas model. Secondly, it enables rule-based question generation using pre-defined question templates, substituting relevant keywords, and defining the corresponding answer column. Furthermore, the toolkit offers options for dataset augmentation through state-of-the-art character or word-based augmentation techniques. The resulting dataset's structure is identical to the format from the sequential QA (SQA) dataset introduced by Iyyer et al. (2016). Section 4.2.2 of this thesis discusses all aspects of the toolkit in more detail.

The maintenance data used in this work comprises 328 maintenance log entries in a tabular structure. Each entry comprises the name of the equipment, the time of failure, a report text, a failure category, a following action, the person who resolved the issue, the respective time, and a contact person. An example can be seen in Table 7.

Table 7 | Maintenance logs which document downevents and taken actions for specific equipment

EQUIPMENT	TIME_STAMP	GUI_BUCHUNGSTEXT	DOWN_EVENT	AKTION	AUTHOR	CREATED	CONTACT_PERSON
Equipment_1	16.01.2018 13:59	Operator_1 temperatur error	ST	TCU resetet und befüllt -> Temp. wieder ok! Person_1	Person_1	16.01.2018 15:30	Contactperson_1
Equipment_2	16.01.2018 13:03	Operator_1 ETCH TIME MAIN ETCH	ST	MHz Generator Reset durchgeführt. Person_2	Person_2	16.01.2018 13:26	Contactperson_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

4.2.1 Pre-processing

Independent of the pre-trained model or fine-tuning method, good results require sufficient data quality, which makes pre-processing of the available dataset necessary. The steps applied include:

1. **Translation to English:** The dataset was translated from German to English, given that most LLMs, including the TaPas model, are pre-trained on English text corpora. Non-English text data typically yields worse performance.
2. **Author abbreviations:** In the report and action columns, author name abbreviations appear next to the text. For the action text, abbreviations at the end were removed, as the "Author" column already contained full names. In the report section, the author names that appeared at the beginning were split and inserted into a separate column.[7]
3. **Date and time separation:** The date and time information in the second column was separated, and the "CREATED" column was deleted to improve clarity.
4. **Header row modification:** Renamed the header row for improved readability.
5. **Handling empty contact person cells:** All empty contact person cells were changed to “None.”
6. **Whitespace removal:** Removal of unnecessary white spaces.

The mentioned steps were implemented using Excel® formulas, and the result can be seen in Table 8.

[7] According to Article 5 of the General Data Protection Regulation (GDPR), personal data (e.g., names, IDs) was anonymized for the purpose of this research.

Table 8 | The cleaned maintenance log dataset after executing all pre-processing steps

equipment	date	time	report	reported by	category	action	solved by	contact person
Equipment_1	16.01.2018	13:59	temperature error	Operator_1	ST	TCU resets and fills -> temp. ok again!	Person_1	Contactperson_1
Equipment_2	16.01.2018	13:03	ETCH TIME MAIN ETCH	Operator_1	ST	MHz generator reset performed.	Person_2	Contactperson_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

4.2.2 Rule-based Question Allocation

As mentioned in Section 4.1, the TaPas model learned text-table correlations during pre-training but requires additional fine-tuning for semantic parsing tasks. Even though the work from (Herzig et al., 2020) already provides several fine-tuned checkpoints, these models have been trained exclusively on publicly available Wikipedia tables, lacking an understanding of domain-specific queries and their corresponding tables. Given the high cost associated with the manual creation of a semantic parsing dataset, this thesis proposes a rule-based question allocation method utilizing pre-defined question templates stored in CSV files. The complete training dataset creation process comprises the following steps:

1. **Importing maintenance logs table:** The maintenance logs table is imported as the primary dataset.
2. **Sub-table splitting:** The input data is split into sub-tables that the TaPas model can process.
3. **Definition of pre-defined questions and keyword-target column dictionary:** Pre-defined questions are defined alongside the creation of a dictionary, mapping keywords to target columns.
4. **Question selection:** Random questions are selected based on the target column.
5. **Keyword replacement**: Keywords in selected questions are replaced to create a unique question.
6. **Answer allocation:** Relevant answers are allocated from the answer column.
7. **Sub-table and dataset export:** Processed sub-tables and the finalized dataset are exported for subsequent fine-tuning.

Table 9 | Sub-table, containing 9 columns and 11 rows, that can be processed by TaPas

equipment	date	time	report	reported by	category	action	solved by	contact person
Equipment_1	16.01.2018	13:59	temperature error	Operator_1	ST	TCU resets and fills -> temp. ok again!	Person_1	Contactperson_1
Equipment_2	16.01.2018	13:03	ETCH TIME MAIN ETCH	Operator_2	ST	MHz generator reset performed.	Person_2	Contactperson_2
Equipment_3	15.01.2018	03:30	transport	Operator_3	ST	retry ok	Person_2	Contactperson_3
Equipment_4	14.01.2018	01:34	T_PROC_ME_V_DEVMED_C ETCH TIME MAIN ETCH	Operator_4			Person_2	Contactperson_4
Equipment_3	13.01.2018	22:01	transport	Operator_3	ST	AL1 open ok	fishric	Contactperson_3
Equipment_3	13.01.2018	17:04	transport	Operator_3	ST	Retry open PM1...ok	Person_1	Contactperson_3
Equipment_5	13.01.2018	13:47	Plant does not start	Operator_5	ST	Los rebooted... ok	Person_1	Contactperson_5
Equipment_3	13.01.2018	12:14	transport	Operator_3	ST	Slit Door AL1 pneumatic hoses reconnected > Open/Close ok!	Person_2	Contactperson_3
Equipment_3	13.01.2018	02:51	transport	Operator_3	ST	AL1 Open/Close, ok	Person_1	Contactperson_3
Equipment_3	10.01.2018	05:01	transport	Operator_3	ST	Slitvalve manually controlled via solenoids. Then repeated open/close via software OK	Person_2	Contactperson_3

In detail, after importing the complete maintenance log table as a CSV file, it should be split to ensure each resulting sub-table is entirely encodable by the TaPas model. Given the model's default maximum sequence length of 512 tokens, the processed maintenance log table, as shown in Table 8, is separated into sub-tables consisting of 9 columns and 11 rows, including the header. Thus, each sub-table comprises a total of 99 cells. A unique identifier is assigned to each sub-table to allow subsequent identification of these sub-tables during the answer allocation process and the table retrieval task within a chatbot framework. An example of a sub-table is depicted in Table 9.

Before questions can be asked about the table and their respective answers can be allocated, a pool of blank template questions, including keywords, which are replaced later, should be defined, as seen in Figure 26.

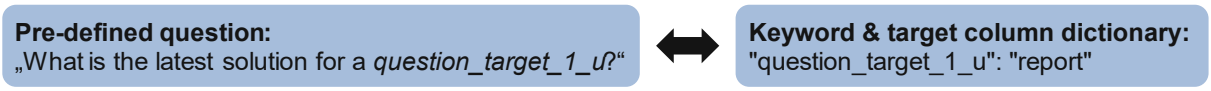


Figure 26 | Definition of pre-defined questions & keyword-target column dictionary

Here, *“What is the latest solution for a question_target_1_u?”* is a question about the “action”-column for a specific report. The relation between the keyword and its target column is captured in a dictionary with the keywords as the key and the target column as its value. The name and number of keywords are arbitrary but must be unique in the question string. However, the suffix *“_u”* of the keyword *“question_target_1_u”* stands for “unique” and specifies the question type. In this case, it means there can only be one unique answer to the pre-defined question. When looking at Table 9, the first table of the sub-table database from the prior step, the report “transport” is listed multiple times. This means there are multiple answers to the same question. To handle multiple occurrences, the rule-based question allocation method of this work provides the following question types with their respective answer:

- **Earliest (Suffix: *“_e”*):** Identifies the earliest occurrence based on the "date" column as the response.
- **Latest (Suffix: *“_l”*):** Identifies the most recent occurrence based on the "date" column as the response.
- **All (Suffix: *“_a”*):** Considers all occurrences as the response cells.
- **Specific (Suffix: *“_s”*):** Requires an extended query incorporating two keywords, one of which is replaced by the precise date of a randomly selected occurrence. Specifies the chosen occurrence as the response cell.

This framework is subsequently employed in the phases of question selection, keyword substitution, and response allocation, detailed in Figure 27. After establishing a question pool, denoted by the name of its corresponding answer column in the header cell, the script iterates through each row of the imported dataset. A question is

randomly chosen for each predetermined question type, and the keyword in the target column is replaced. The resulting question, table path, answer position, and response text are consolidated and stored as an entry in the output table.

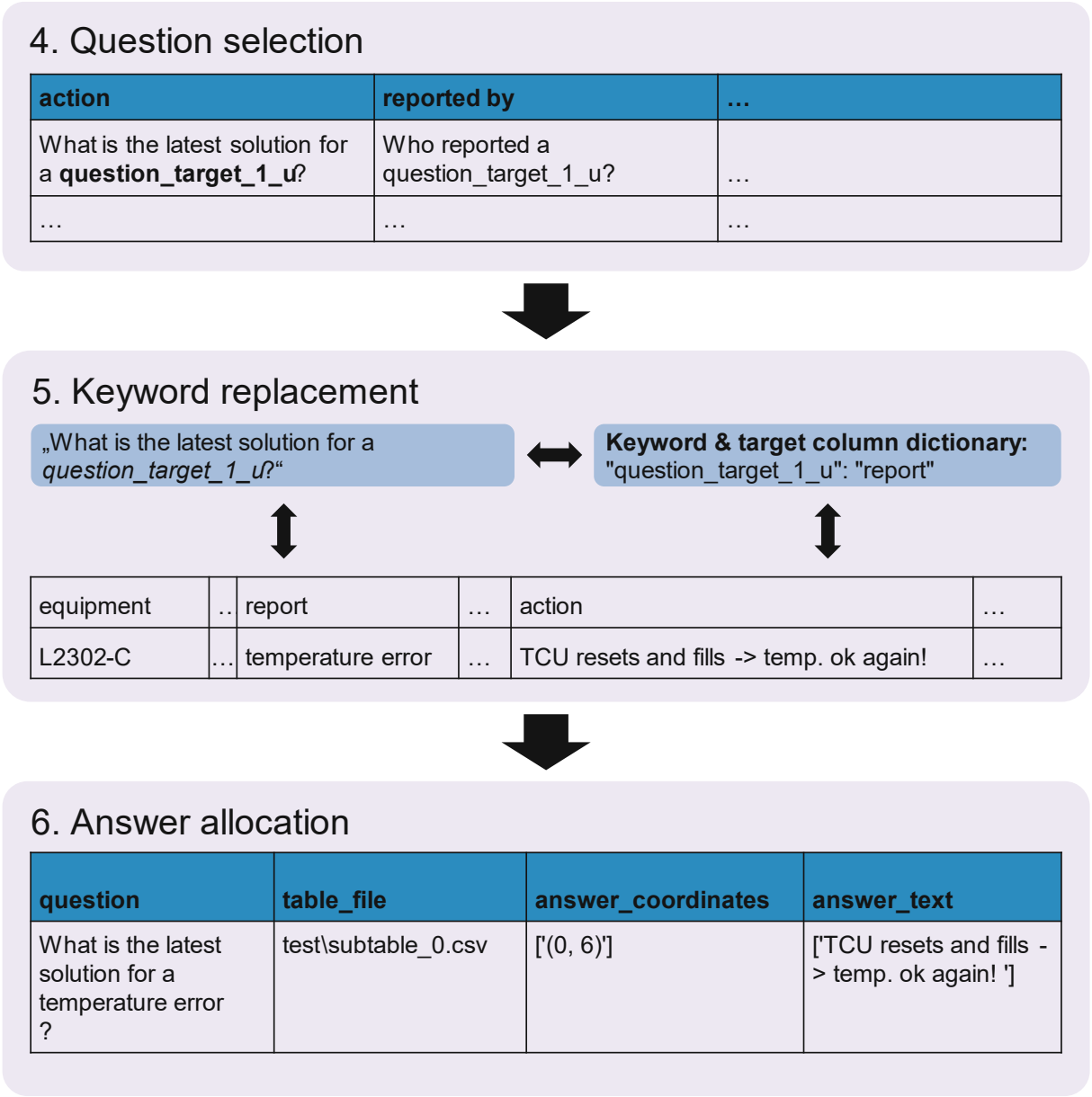


Figure 27 | Question selection, keyword replacement, and answer allocation steps, executed during the maintenance dataset crafting process

Subsequently, the generated training table is split into training and testing subsets before being exported with the sub-table database. These saved files can now be used directly for fine-tuning the TaPas model, enabling the understanding of maintenance-specific queries. The fine-tuning method, which was deployed in this work, will be detailed in section 4.3.

4.2.3 Domain-specific Data Augmentation

The rule-based question allocation method described in the prior section generates 1,088 question-answer pairs originating from pre-determined questions for four

columns of the maintenance log table. This dataset is comparatively small relative to the datasets employed in question-answering tasks discussed in the systematic literature review of Section 3.2. This may eventually lead to insufficient predictions of the final model after the fine-tuning process since the performance of DL models generally improves with the size and diversity of its training set. Consequently, addressing the issue of data scarcity becomes necessary, and this can be effectively managed through data augmentation techniques.

Data augmentation is a widely utilized methodology in computer vision. However, it has gained increased attention in the NLP research community due to the increased interest in LLMs and work in low-resource domains (Feng et al., 2021). In general, augmentation can be achieved by incorporating modified copies of existing data or generating entirely new data based on the existing dataset (B. Li et al., 2022). These techniques can be broadly categorized into three groups according to B. Li et al. (2022):

- **Paraphrasing methods:** They generate augmented data with limited semantic difference from the original data.
- **Noising methods:** They add discrete or continuous noise.
- **Sampling methods:** They master the data distribution of the original set and sample new data within it to increase diversity.

Figure 28 illustrates that these methods enhance diversity while maintaining varying degrees of semantic similarity with the original input. Paraphrasing methods operate at different levels, such as word, phrase, and sentence levels, utilizing resources like thesauruses, semantic embeddings, or language models. They generate semantically similar sentences to the original data (B. Li et al., 2022). On the other hand, noising methods introduce modifications to the original data without significantly altering the sentence's overall semantics. This can involve character and word swapping, deletion, insertion, or substitution. The primary objective is to improve the overall robustness of trained models (B. Li et al., 2022). Finally, sampling methods master the original dataset's data distribution and draw new data samples from it, thereby increasing diversity. Unlike paraphrasing methods, which only require original sentences as input, sampling methods require task-specific information such as labels and data format (B. Li et al., 2022).

Feng et al. (2021) found that operating within specialized domains, like the industrial maintenance industry, characterized by domain-specific vocabulary and jargon, poses significant challenges. Pre-trained models and external knowledge resources for data augmentation, such as WordNet, often prove ineffective in these contexts. This reduction in effectiveness is attributed to the significant difference between the distribution of augmented data and the original data (Feng et al., 2021).

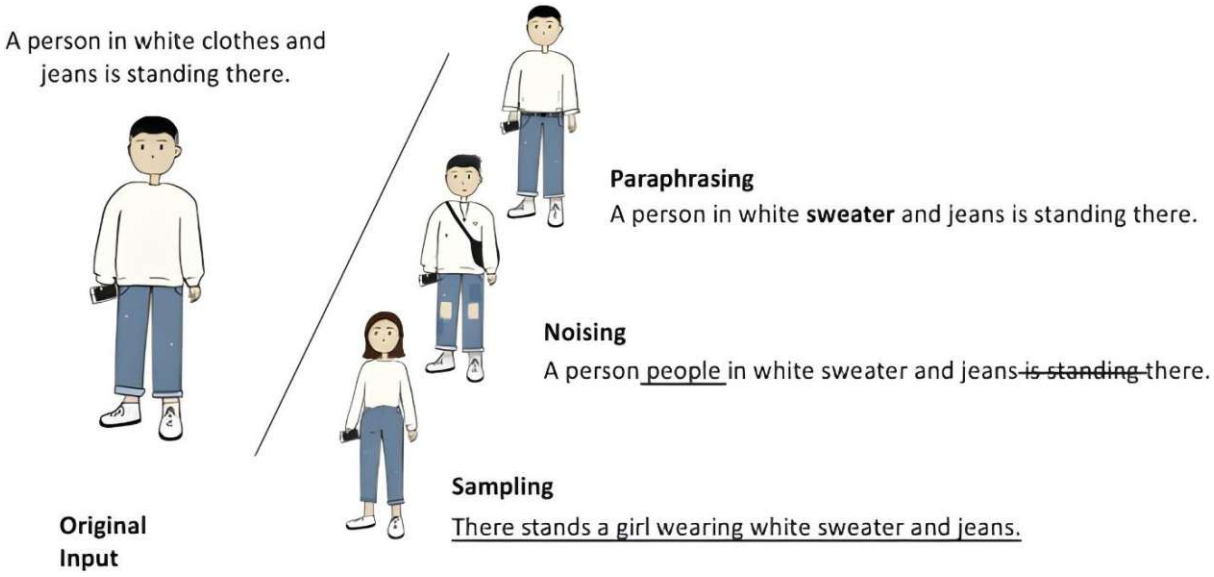


Figure 28 | Data augmentation techniques increase the diversity of the original dataset by adding modified copies or creating entirely new datasets (B. Li et al., 2022)

To the author's knowledge, no domain-specific data augmentation approach in the context of NLP tailored to the industrial maintenance industry exists. Hence, a domain-specific approach should be developed that addresses the challenges of the maintenance domain in the context of the crafted table question-answering dataset from Section 4.2.2. Those challenges are:

- **Short questions:** The questions that need to be augmented comprise only a single sentence.
- **Missing context:** There is no context to a question or answer in the form of a paragraph. All entries in the maintenance log database are unrelated.
- **Domain-specific language:** Most answers contain domain-specific language.

Given the unknown underlying data distribution of the question-answering dataset introduced in the previous section, this thesis focuses solely on paraphrasing and noising methods for data augmentation. Notably, augmenting single-sentence questions presents a significant challenge due to the absence of contextual information from paragraphs. Unnecessary noise introduced at the word or character level may result in the loss of meaning for a sentence or question, potentially affecting the performance of the fine-tuned model negatively. This challenge also extends to paraphrasing methods. An example is shown in Figure 29. Altering the sentence *"What is the latest solution for a temperature error?"* to *"What is the earliest solution for a temperature error?"* results in semantically similar questions. However, they demand entirely different answers, especially if multiple entries for a temperature error exist in the dataset. Additionally, the right answer cannot be derived from the surrounding context. Hence, a balance between augmentation and maintainability of semantic meaning must be reached to achieve optimal fine-tuning outcomes.

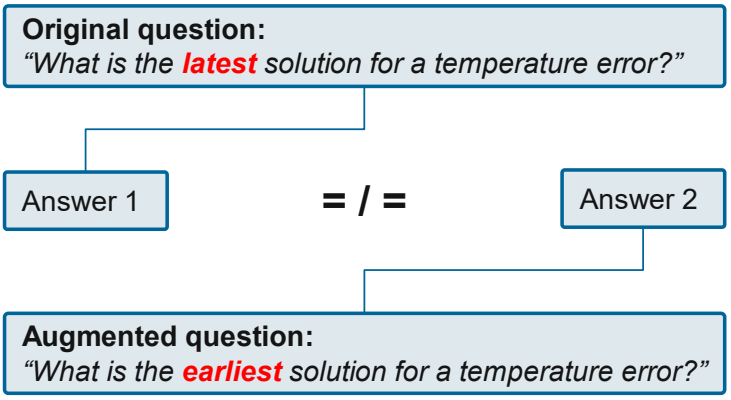


Figure 29 | Distortion in single-sentence questions without any context may lead to the wrong allocation of answers

The augmentation techniques are incorporated into the dataset crafting library from Section 4.2.2, using the publicly available library by Ma (2019), which provides state-of-the-art NLP data augmentation methods. Three different augmented datasets have been created using the following methods:

- **Character-based augmentation:** Keyboard errors
- **Word-based augmentation:** Contextual word embeddings
- **Word-based augmentation:** Back translation

The first one simulates typing errors based on keyboard distance. Here, characters are inserted in a maximum of two words for each sentence. This does not include special characters or numeric values to maintain the original meaning of each question. Even though humans can reduce the impact of inserted noise through their prior knowledge and semantic understanding, it can pose a significant challenge for language models, potentially improving their robustness after the training process. For the word-based augmentation, two approaches have been used. First, one random word of each question is paraphrased by feeding its surrounding words from the same question to a BERT model ("bert-base-uncased") to find the best possible substitution. This method considers the context of the word to be substituted and limits the effect on the original sentence semantics. It also omits the requirement of context surrounding a question. The second method paraphrases the whole sentence on a word level by translating it into a defined target language (German) and translating it back to its source language (English), using translation models ("facebook/wmt19-en-de"). It ensures correct syntax and unchanged semantics. However, it lacks customizability due to the fixed language model (B. Li et al., 2022).

Each augmentation method is applied only to the original questions, keeping the corresponding tables' content unchanged. This simulates the behavior of real-world users, who generally do not know the content of maintenance logs and, therefore, paraphrase or misspell specific terms, increasing the question diversity and potentially improving the model's accuracy and robustness after the fine-tuning process.

4.3 Fine-tuning Method

In order to adapt TaPas to the specific requirements of the industrial maintenance domain, it is fine-tuned using the generated dataset from Section 4.2.2 and the augmented datasets from Section 4.2.3. The train test split for all sets corresponds to 80% training and 20% test data, on which the resulting models are evaluated. The fine-tuning method is identical to the original paper from Herzig et al. (2020) by adding a randomly initialized cell selection head on top of the pre-trained base model and then training it using the pipeline described by Rogge (2023). The process begins using the publicly available checkpoint on the Huggingface-Hub “Tapas-base” to achieve a good trade-off between accuracy and practicality regarding its model size. Additionally, the checkpoint “Tapas-base-finetuned-sqa” is used, which was already fine-tuned on the SQA dataset, to evaluate if the model profits from prior non-domain-specific knowledge.

The fine-tuning process has been conducted natively in PyTorch, employing the conventional BERT tokenizer for processing questions, table cells, and headers while maintaining a maximum sequence length constraint of 512 tokens. The adaptive learning rate algorithm, specifically the Adam optimizer, has been leveraged in this work and was executed on a single GPU. For optimal training results, the following hyperparameters must be tuned:

- **Batch size:** A hyperparameter that defines the number of input data rows that must be processed before updating the internal model parameters (Brownlee, 2022).
- **Learning rate:** The initial proportion that internal model weights are updated between each batch. Larger learning rates result in faster initial learning. Smaller values slow learning down during the training process (Brownlee, 2021)
- **Epochs:** A hyperparameter that defines the number of times the learning algorithm will process the entire training dataset (Brownlee, 2022).

Generally, Adam’s learning rate is gradually reduced to improve the model’s final performance on the test set. However, Smith et al. (2017) found that increasing batch size obtains the same learning curves. Therefore, to determine the optimal hyperparameters, a grid search approach for batch size selection has been employed in combination with two different learning rates. The search commenced with a batch size of 2 and iteratively increased until the model's performance on the development set started to drop. This resulted in six combinations of hyperparameters, which are concluded in Table 10. To ensure stable training, the number of training epochs was fixed at 3. Using this setup, fine-tuning took a total of 9 hours to complete. The training results and performance on the development set will be discussed in Section 5.

Table 10 | Overview of hyperparameters for fine-tuning the TaPas model

Parameter	Values
Learning rate	(2e-5, 5e-5)
Batch size	(2, 4, 8)
Epochs	(3)

4.4 Chatbot Architecture

As mentioned at the beginning of Section 4, Rasa comprises an NLU module and a dialog management module as its main components. Its architecture is further detailed in Figure 30, in which the NLU module is represented by the NLU pipeline and the dialog management module by the dialogue policies.

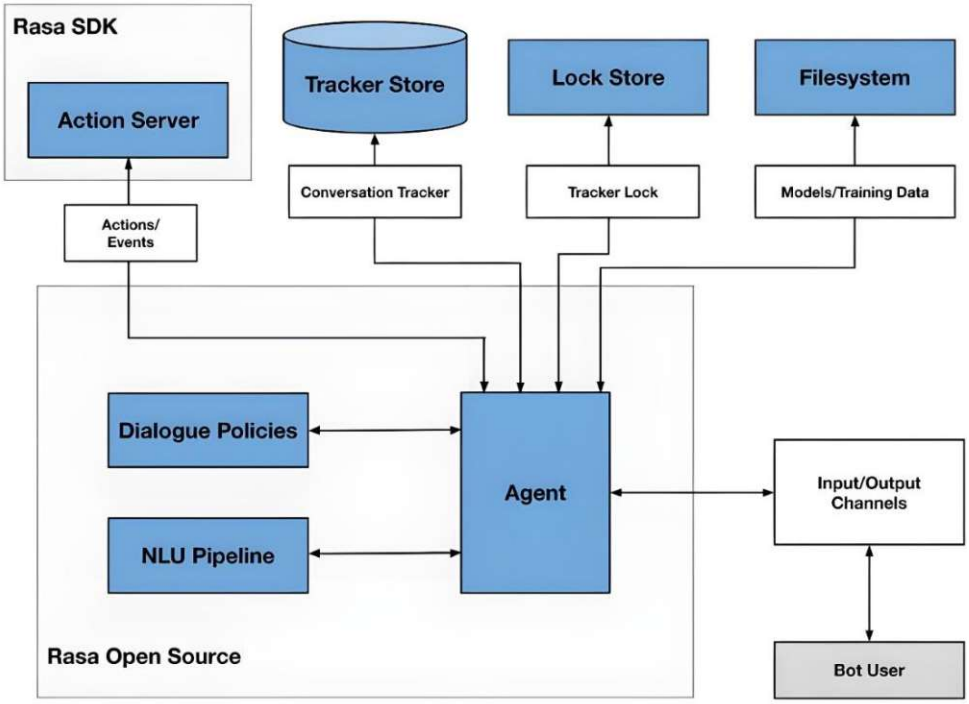


Figure 30 | General architecture of a Rasa Open Source chatbot (Rasa Technologies, 2023b)

Another core feature of Rasa is the Rasa software development kit (SDK), which allows running custom actions on a separate server instance called an action server (Rasa Technologies, 2023b). This allows the implementation of the developed table question-answering pipeline based on the fine-tuned TaPas model into an existing rasa framework.

Furthermore, ancillary modules support Rasa's capabilities. The lock store ensures the orderly handling of multiple concurrent conversations, safeguarding data integrity, while the tracker store tracks the historical context of interactions, enriching the contextual understanding of the chatbot (Rasa Technologies, 2023b).

All relevant models, training data, and configuration files for setting up the chatbot are stored in its filesystem, with its structure depicted in Figure 31. In this context, *actions.py* holds the code for the custom actions that run on the custom actions server and can be called by the conversation agent. The *config.yml* file defines the components, such as tokenizers, classifiers, and policies, that the model uses to predict the next actions based on the user's input. The authentication credentials for the chat platform are stored in the *credentials.yml* file services (Rasa Technologies, 2023a).

actions
- *__init__.py*
- *actions.py*

config.yml
credentials.yml

data
- *nlu.yml*
- *rules.yml*
- *stories.yml*

domain.yml
endpoints.yml

models
- *<custom name>.tar.gz*

tests
- *test_stories.yml*

Figure 31 | The default filesystem of a Rasa chatbot

The data folder stores all relevant training data and forms the bedrock of the chatbot's interactions with users. First, the *nlu.yml* file stores information about potentially incoming user messages in a structured way, one of which is the user's intent and extracted entities. Afterward, to execute a specific action for different user's messages, dialog rules can be defined in the *rules.yml* file that links a specific intent to its respective action. In this fashion, small pieces of dialog can be defined. However, the chatbot cannot generalize between unseen user messages. Thus, stories, representations of conversational scenarios between a user and a bot, can be defined in the *stories.yml* file, which realized longer conversation paths services (Rasa Technologies, 2023a).

Besides, the *domain.yml* consolidates all defined intents, entities, responses, and actions the chatbot can use. The *endpoints.yml* file defines all communication endpoints through which the bot can communicate with external systems and services (Rasa Technologies, 2023a).

To establish a proficient chatbot catering efficiently to users' requirements within the industrial maintenance sector, the files underwent customization, detailed in

Subsections 4.4.1 to 4.4.4. These modifications were undertaken to align the chatbot's functionalities with the specific demands of the industry.

4.4.1 Intent Recognition

The chatbot's dialog structure is based on a standard maintenance process that can describe all possible maintenance tasks, ranging from simple to complex repairs. Matyas (2022b) defined such a standard maintenance process in 8 steps to achieve a clear separation between value-adding and support process steps. The process template is depicted in Figure 32 as a predecessor-successor relationship and consists of the following steps:

1. Trigger
2. Work preparation (WP) planning
3. WP execution
4. Manual execution
5. Recommissioning
6. Function check
7. Release
8. Closure

Additionally, documentation is added as a supporting process throughout all eight steps. It starts with a trigger event, which can be the occurrence of a malfunction or planned maintenance activity and initiates the mean time to repair (MTTR). Its objective is to systematically gather all necessary information essential for the efficient planning and execution of maintenance tasks. Subsequently, appropriate measures are deduced based on this information. If necessary, an inspection is conducted as a part of this process. Afterward, WP planning and WP execution follow, which encompasses the allocation of personnel, materials, essential equipment, and the determination of the execution date. Subsequently, the systematic procurement of the planned resources, including materials, equipment, tools, and spare parts, follows. Simultaneously, the designated area undergoes clearance of obstructive elements, and any obtrusive components are disassembled (Matyas, 2022b).

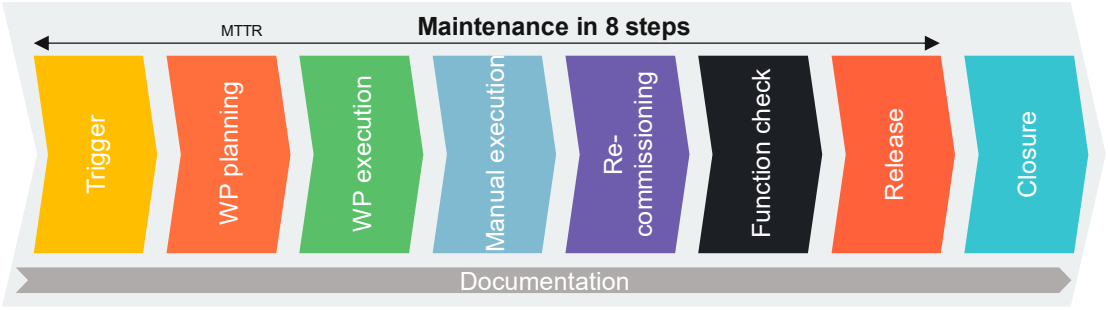


Figure 32 | Standardized 8 step maintenance process able to describe simple and complex maintenance tasks (Matyas, 2022b)

Step 4 comprises executing the planned maintenance tasks, deducted from step 1. It is followed by a recommissioning step that includes the quality assessment of the plant, building, and safety-related components and the commissioning itself. Afterward, the quality of the performed tasks must be checked during the function check step and entered into the maintenance logs. To complete the maintenance process, the equipment is handed over again into the responsibility of the user, and the completion of the task is reported in the last step closure (Matyas, 2022b).

Based on this description, it is now possible to define possible user inputs and, therefore, the *nlu.yml* training file, which is laid out in Figure 33 and serves later as a basis for the dialog management system.

```yaml
version: "3.1"
nlu:
- intent: greet
  examples: |
    - hey
    - hello
    - hi
- intent: goodbye
  examples: |
    - good afternoon
    - cu
    - good by
- intent: affirm
  examples: |
    - yes
    - y
    - indeed
- intent: deny
  examples: |
    - no
    - n
    - never
- intent: start_maintenance
  examples: |
    - I am ready to start the maintenance process.
    - Thank you, i will start.
    - OK, start.
- intent: table_qa
  examples: |
    - What to do against a temperature error?
    - Who was the author of the temperature error?
    - Who solved the temperature error?
- intent: general_questions
  examples: |
    - Explain to me how a torque wrench works!
    - What is a metric bolt?
    - What is a sealing ring?
- intent: work_done
  examples: |
    - I am done!
    - Done
    - Ready
```

Figure 33 | The *nlu.yml* file builds the foundation for RASA's dialog management system

The initial interaction between the user and the chatbot typically entails a greeting, thereby establishing an intent labeled as *greet*. This intent encompasses various user inputs, including expressions like "hey," "hello," and "hi." Subsequently, users may seek information regarding reported incidents during the work preparation phase, denoted as *table_qa*, which eventually will call the fine-tuned TaPas model for executing the QA task on the maintenance log database. Following this, users might express an intention to progress from step 2 (WP planning) to steps 3 and 4 (WP execution and manual execution). This intent is labeled as *start_maintenance*. As the interaction continues, users may convey information regarding completed tasks, categorized under the intent *work_done*. Furthermore, the chatbot should be equipped to address general inquiries posed by users, denoted as the intent *general_questions*. Afterward, the user informs the chatbot about the completed work (*work_done*). Eventually, users may affirm or negate specific queries posed by the chatbot before concluding the conversation.

Leveraging the data presented in Figure 33, Rasa's NLU pipeline can be trained to predict the user's intent. It must be noted that for the sake of brevity and readability, not the complete set of examples used for training of the industrial maintenance chatbot is displayed in Figure 33.

In this thesis, the recommended pipeline without pre-trained word embeddings for better adaptation to domain-specific terminology is configured in the *config.yml* file and depicted in Figure 34. Only the threshold of the *FallbackClassifier* was changed to 0,7. This means that if the model's prediction of a user intent is lower than the specified threshold, it will trigger the default intent *nlu_fallback*. Subsequent to the construction of the *nlu.yml* file, it serves as the foundation upon which dialog stories and rules are formulated to facilitate smooth dialog management.

```yaml
language: en

pipeline:
- name: WhitespaceTokenizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier
  entity_recognition: false
  epochs: 100
- name: RegexEntityExtractor
  use_lookup_tables: false
- name: FallbackClassifier
  threshold: 0.7
```

Figure 34 | The *config.yml* specifies the components of the NLU and NLP pipeline

4.4.2 Dialog Management

Until now, the NLU module of the chatbot can only predict the user's intent but does not know how to react. Thus, a dialog that resembles a possible conversation path in the industrial maintenance industry should be defined. An exemplary dialog can be seen in Figure 35. It links an intent to a specific action, which can be a pre-defined response from Figure 36 or the execution of a custom action, discussed in detail in the following sections.

```
version: "2.0"

stories:

- story: interactive_story_1
  steps:
  - intent: greet
  - action: utter_greet
  - intent: table_qa
  - action: action_table_qa
  - intent: table_qa
  - action: action_table_qa
  - intent: start_maintenance
  - action: utter_alert
  - action: utter_ama
  - intent: table_qa
  - action: action_table_qa
  - action: utter_check_if_ready
  - intent: work_done
  - action: utter_function
  - intent: affirm
  - action: utter_release
  - intent: affirm
  - action: utter_release_confirm
```

Figure 35 | Stories in the *stories.yml* file link intents to actions

As in the preceding section, the dialog is based on the standardized 8-step maintenance process defined by Matyas (2022b) and starts with a greeting of the user, which is returned by a greeting (*utter_greet*) of the bot. This part includes information about the latest reported downtime events. This event represents the trigger, and the user wants to gather information for the planning and execution of the maintenance task. Thus, the intent for asking questions about the maintenance log database (*table_qa*) and the respective action, calling the fine-tuned TaPas model (*action_table_qa*), is implemented twice. As described in the prior section, the user wants to start the maintenance process afterward with the intent *start_maintenance*. Subsequently, the chatbot alerts the user with *utter_alert* to check if the WP preparation and execution (steps 2 and 3) have been done successfully and it is safe to proceed to the manual execution (step 4). Additionally, the bot hints that the user can keep asking questions during the maintenance process (*utter_ama*), to which the user responds with another question about the maintenance logs database (*table_qa* and *action_table_qa*). After giving the user a response, the bot reminds the user to give a

sign when finished (*utter_check_if_ready*), and the user responds with the intent *work_done*. To complete the 8-step maintenance process, the chatbot asks about the functionality (*utter_function*) and if the equipment should be released again to the production line (*utter_release*), which the user confirms. Finally, the chatbot tells the user that the equipment has been released to the production line (*utter_release_confirm*), and the interaction concludes.

```
responses:
  utter_greet:
  - text: |-
      Hello! There are currently three open issues:
       - 1. Radiator conversion + WAP adjustment
       - 2. Equipment_1 cannot be operated from the system screen FRONT
       - 3. shows wafers where none is!?!?!?
  utter_alert:
  - text: Remember to make sure the machine is turned off and the work
area is safe before starting work ⚠.
  utter_ama:
  - text: If you have any question regarding your maintenance task, ask
me!
  - text: Should you require any information or assistance with your
maintenance task, feel free to inquire here!
  - text: If there's anything you'd like to know about your maintenance
task, don't hesitate to ask, and I'll provide the answers you need.
  utter_check_if_ready:
  - text: Tell me when you are finished with your maintenance task ˣ◌ـ!
  utter_function:
  - text: Now please perform a manual function test and a test run with
reduced power. Is the machine working again?
  - text: Now please perform a manual function test and a test run with
reduced power. Has the issue with the machine been resolved?
  - text: Now please perform a manual function test and a test run with
reduced power. Is the device functioning as expected now?
  utter_release:
  - text: Great! Do you want to release to machine to the production
line and end the maintenance task?
  - text: Percfect! Would you like to put the machine back into
production and conclude the maintenance process?
  - text: Great! Shall we return the machine to the production line and
mark the maintenance task as completed?
  utter_release_confirm:
  - text: Alright! The machine is released to the production line and
the maintence task got documented.
  utter_goodbye:
  - text: Bye
```

Figure 36 | The *domain.yml* file defines all possible actions of the maintenance chatbot

For some user intents, it is advantageous to force a specific action by defining conversation rules in the *rules.yml* file. As seen in Figure 37, the intent *greet* always forces *utter_greet* by the chatbot. This ensures that the user can always start the conversion. Similarly, saying goodbye forces the chatbot to say goodbye and reset the conversation history to avoid getting stuck and having to restart the chatbot system entirely. User intents that are out-of-scope (*nlu_fallback*) or general questions that are not part of the previously defined dialogue story should always be handled by a custom

action and not a pre-defined answer and will be discussed in the following subsections.

```
rules:
- rule: Greet anytime the user greets
  steps:
  - intent: greet
  - action: utter_greet

- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: utter_goodbye
  - action: action_restart

- rule: Fallback answer
  steps:
  - intent: nlu_fallback
  - action: action_nlu_fallback

- rule: General questions
  steps:
  - intent: general_questions
  - action: action_nlu_fallback
```

Figure 37 | Rules force specific action inside the RASA framework during a conversation

4.4.3 Maintenance Log Question Answering

The first custom action that is implemented in Rasa is the maintenance log QA pipeline, which lets the user ask domain-specific questions about industrial maintenance tasks and is the main focus of this thesis. It is visualized in Figure 38 and contains three steps: information retrieval, cell selection, and answer generation.

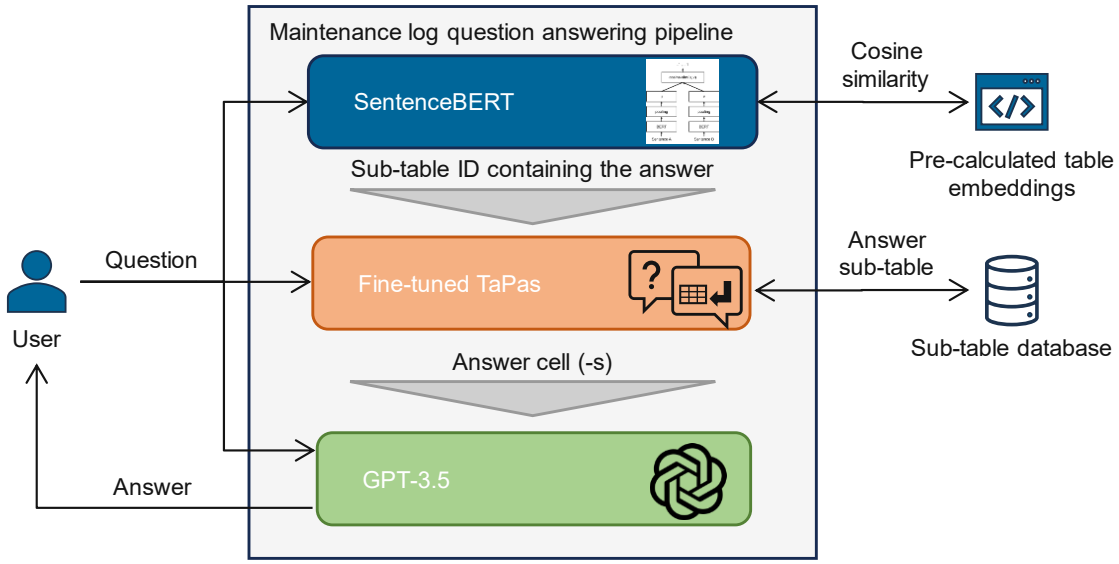


Figure 38 | The maintenance log question answering pipeline is based on three connected LLMs

Since TaPas has only a restricted encoding length, the source maintenance log table is split into a sub-table database, which can be handled by the model (as described in Section 4.2). Its drawback is that TaPas now relies on encoding a table containing the answer to the user's question to select the correct answer cell (-s).

Question:	*For which equipment was shows wafers where none are reported?*

Rank	Score	Flattened table row
1	0.6555	08:45,shows wafers where none is!?!?!? EdcID:XXXXXXXXXX,Operator_1,Deleted wafer in Material Editor. ,test_dev\subtable_29.csv
2	0.5533	10:14,"T_ME_V EQUIPMENT:Equipment_1,LOT:XXXXXXXXXX, RECIPE:XXXXXX, ANNEX TO IHXXXXXXXXX; ETCH TIME DEVIATION",APCFDC:,APC alarm due to process abort (He Flow). He flow at next wafer ok , test_dev\subtable_28.csv
3	0.5509	23:08,flow EdcID:XXXXXXXXXXX,Operator_2,Wafer ,test_dev\subtable_29.csv

Figure 39 | SentenceBERT retrieves sub-tables by computing the semantic similarity between the user's query and table row

Thus, an information retrieval component has been implemented in this thesis using the pre-trained SentenceBERT model. As described in Section 3.2.3, SentenceBERT leverages siamese and triplet network architectures to compute the semantic similarity between sentences efficiently using their cosine similarity (Reimers & Gurevych, 2019). To retrieve the most similar table to a user's question, each sub-table from the database is flattened with each row encoded separately, including an additional "ID"-column, using the "deepset_all-mpnet-base-v2-table" model. It is designed as a general-purpose model and scores the highest average performance in sentence embedding and semantic search tasks (Reimers, 2022). The calculated embeddings are then stored as a separate file, which is loaded when the Rasa server is started for improved computational efficiency. Re-encoding is, therefore, only necessary if the maintenance log database is extended. When Rasa's dialog management system triggers the maintenance log QA pipeline, the user's question is encoded and matched against the table embeddings, and their cosine similarity is computed. An example can be seen in Figure 39. Here, the top three results and their respective score are shown for the user's question: *"For which equipment was shows wafers where none are reported?".* Even though the top result is the correct answer, its cosine similarity score is only 0.66, which is a result of encoding each flattened row entirely. This not only adds irrelevant data, like time, author, action, and sub-table ID, but drastically increases the total length of the corpus in comparison to the user's question. This might be improved by using models suitable for asymmetric search, like "Msmarco-MiniLM-L-6-v3", which are able to find longer answer passages for short questions (Reimers, 2022). However, for most entries in the maintenance log database in this thesis, "Deepset_all-mpnet-base-v2-table" provides sufficient results.

The sub-table ID of the top-scoring entry is extracted, and the whole table and the user's question are tokenized and fed into the fine-tuned TaPas model, as described in Section 4.1. The TaPas model and its tokenizer are stored locally with the same

configuration used during fine-tuning (see Section 4.3) to avoid downloading them each time the Rasa dialog manager calls the custom action. After inference, there are three possible outputs of TaPas, described in Figure 40. In the case of a single cell selection, the cell content is directly converted into a string of text. However, for multiple cell selection, the content of the cells is converted to a string and numbered for better readability. Finally, if no cell is selected, TaPas returns an empty string.

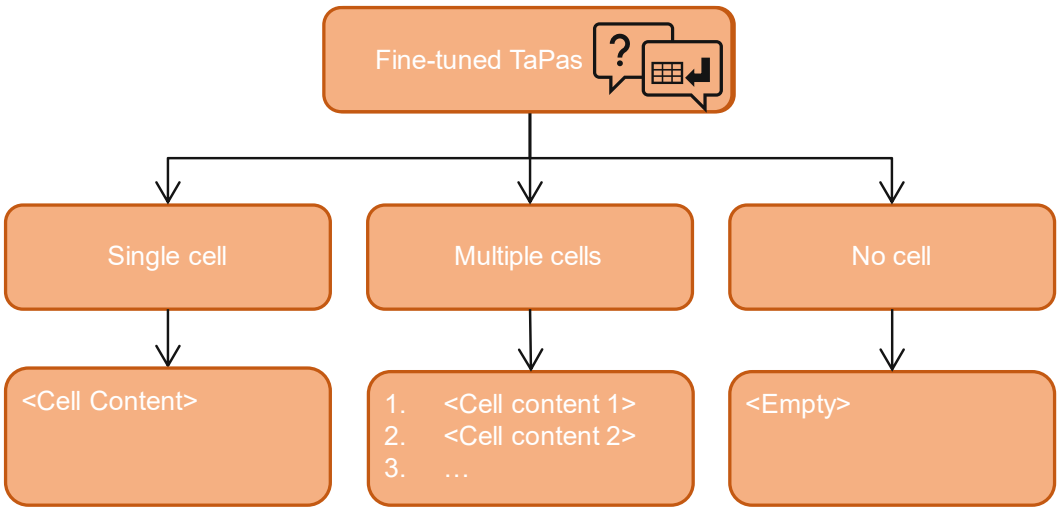


Figure 40 | Answer structure of the fine-tuned TaPas model

As an encoder-only model, TaPas can only perform extractive QA tasks, which means finding an answer to a given question in a longer text corpus, such as a paragraph or, in this case, a table, and returning it regardless of how the question was phrased. Therefore, a generative third layer is added to the maintenance log QA pipeline to improve the understandability of its output. This is achieved by inserting the output of the TaPas model into a pre-defined prompt and feeding it to a GPT-3.5 pipeline, which can be seen in Figure 41. It is worth noting that models like GPT are prone to hallucinations, which might reduce the accuracy of the answer or even make up facts if the output of the TaPas model is unclear or was not included in its pre-training. The GPT-3.5 model is advised to act as a maintenance chatbot and answer the question using the provided answer by paraphrasing it and including the question itself. Combining both the question and the rephrased answer makes the final output more readable to the user, as demonstrated in Figure 42. The correct answer to the question *"What are the steps for Radiator conversion + WAP adjustment?"* and the theoretical output of the TaPas model is denoted in the middle and can be hard to comprehend without expert knowledge. The combined answer of both models overcomes this issue by using whole sentences instead of word pieces and removing typing errors. Implementing this approach into a Rasa framework as a custom action ensures the user receives accurate and comprehensible answers based on the maintenance log database.

Prompt for generating a final answer

Act as a maintenance chatbot and answer the following question, using the already provided answer. Answer in a complete sentence by paraphrasing the answer provided and including the question.

Answer: {Cells returned by the fine -tuned TaPas model}

Question: {User's question}

Figure 41 | Prompt for GPT-3.5 to generate a user-oriented answer from the TaPas input

Example of Radiator conversion + WAP adjustment

Question: What are the steps for Radiator conversion + WAP adjustment?

TaPas: 1. HW ANko after chamber opening OK --> Anko required --> then Engineering . 2. chamber open. Checked WAP rings and gap. WAP conductance test with new setup parameters and 1.52mm WAP mechanism OK Wait LR Operator_1. . 3. WAP Conductance with Two-Piece Confinement Ring config not . 4. Original cooler installed, Wap mech. to 1.52 just. config changed, system pumps for .

TaPas + GPT- 3.5: The steps for Radiator conversion and WAP adjustment include: Anko requirement after chamber opening, followed by Engineering involvement, checking WAP rings and gap after chamber opening, performing a WAP conductance test with new setup parameters and a 1.52mm WAP mechanism, waiting for LR Operator_1, and considering the WAP conductance with a Two-Piece Confinement Ring configuration. Additionally, the original cooler is installed and the WAP mechanism is adjusted to 1.52, with a change in configuration, and the system is pumped for operation.

Figure 42 | The combination of a TaPas and GPT-3.5 model generates user-oriented answers while maintain high answer accuracy

4.4.4 General and Out-of-scope Question Answering

Besides questions about domain-specific maintenance data, the user might ask general questions about maintenance and equipment. Instead of only providing pre-defined answers in the *domain.yml* file, this case is handled gracefully as well by defining an additional custom action. Analog to the maintenance log QA action, the user's question is fed into a pre-defined prompt, which is processed by the same GPT-3.5 pipeline from the proceeding section. The prompt is displayed in Figure 43 and advises the model to act as a maintenance chatbot and use technical details in its response when answering the user's question. However, this time, no additional context is provided, like the answer from the TaPas model. Thus, the answer generation only depends on the prompt and pre-training of the GPT-3.5 model. The Rasa model is trained to recognize general questions by providing possible examples in the *nlu.yml* file. An example of the pipeline's output is provided in Figure 44, which shows the detailed answer to the question, *"Tell me how to use a torque wrench!"*. It can be seen that the answer returned is, in most cases, sufficient to serve as a general guideline for performing a maintenance task without any additional fine-tuning.

Prompt for generating a general or out-of-scope answer

Act as a maintenance chatbot and answer the following question.
Answer in a complete sentence and use technical details.

Question: {User's question}

Figure 43 | GPT-3.5 is used for generating user-oriented answers for general and out-of-scope questions

Example of a response to general questions

Question: Tell me how to use a torque wrench!

GPT-3.5: To use a torque wrench, begin by setting the desired torque value on the wrench's torque adjustment mechanism. Then, insert the appropriate socket onto the wrench's square drive end and place it onto the fastener you wish to tighten. Apply force in a slow and controlled manner until you feel the wrench click or hear an audible signal indicating that the specified torque has been reached. At this point, stop applying force to avoid over tightening the fastener and potentially causing damage.

Figure 44 | Example of the industrial maintenance chatbot's response to a general user question

Out-of-scope QA is handled by the same custom action but does not rely on a specified intent in the *nlu.yml* file. Instead, the action is triggered by the *FallbackClassifier*, which was defined in Section 4.4.1. Since general and out-of-scope questions might be asked at any time during the conversation between the user and the chatbot, the additional output *UserUtteranceReverted* is passed from the custom actions function. This default RASA function removes the previous user input from Rasa's tracker and avoids unintended deviations from the defined story paths. An example of such an answer can be seen in Figure 45.

Example of a response to out-of-scope questions

Question: Is this a chatbot?

GPT-3.5: Yes, this is a chatbot. I am an AI-powered conversational agent designed to fulfill maintenance-related tasks and provide assistance.

Figure 45 | Example of the industrial maintenance chatbot's response to out-of-scope user questions

5 Evaluation

The evaluation of the developed maintenance chatbot encompasses two distinct phases. Firstly, an assessment of the fine-tuning outcomes for the maintenance TaPas model is detailed in Section 4.3. This evaluation aims to measure the model's adaptation to the industrial maintenance domain through transfer learning. Thereby, it determines the QA system's accuracy and indicates which model should be incorporated into the chatbot framework. In this context, the efficacy of the proposed domain-specific data augmentation methods concerning model performance is addressed and analyzed.

Secondly, a comprehensive evaluation of the entire developed maintenance chatbot application is conducted. To accomplish this, a structured study is executed using the PSSUQ. The PSSUQ is specifically designed to assess the usability and user satisfaction levels associated with software applications, providing valuable insights into the chatbot's practical utility and user experience.

5.1 Maintenance Question Answering Model Accuracy

This section presents the findings of the TaPas fine-tuning process conducted in Section 4.3, using an industrial maintenance log QA data set. For extractive QA, the two most popular evaluation metrics are:

- Exact match
- F_1-score

The Exact Match metric is defined as (Hugging Face, 2023b):

$$E_m = \frac{Number\ of\ correct\ Answers}{Total\ number\ of\ Answers}$$

The exact match score E_m only ranges from 1 if all characters of the predicted answer are identical to the correct answer to 0 if the model predicts any deviating character combination. In contrast, the F_1-score is less stringent, quantifying the harmonic mean of precision and recall (Tunstall et al., 2022). The general definition of the metric is:

$$F_1 = \frac{2 * Prection * Recall}{Precision + Recall}$$

$$\text{with } Precision = \frac{tp}{tp+fp}$$

$$\text{and } Recall = \frac{tp}{tp+fn}$$

Here, true positives tp represent the characters shared between the correct answer and the predicted one, whereas false positives fp denote characters present in the

prediction but absent in the correct answer. Conversely, false negatives fn signify characters absent in the prediction but included in the correct answer. Notably, the F_1-score, while accommodating, is unsuitable for evaluating table QA tasks due to the nature of the model's predictions, which are cells as opposed to text spans in conventional QA tasks. In table-based QA, the correct or incorrect selection of cells defines the answer, necessitating the adoption of the exact match score as a metric in this study.

The fine-tuning outcomes for both model checkpoints, obtained through the grid search method employed to identify optimal hyperparameters, are presented in Figure 46. Notably, the accuracy of answers for both models experienced a decline when the batch size was increased from 4 to 8, regardless of the learning rate, which was established as the termination criterion for the search procedure. However, a slight increase in accuracy is observed for the "Tapas-base" model when the learning rate is reduced, following the drop observed between batch sizes 2 and 4. The best-performing model on the test set emerges as the "Tapas-base-finetuned-sqa," trained with a batch size of 4 and a learning rate of 5e-5, attaining an exact match score of 0.95. In contrast, the best-performing "Tapas-base" model is trained with a batch size of 2 and a learning rate of 2e-5, achieving a score of 0.91. Relative to the accuracy of the non-fine-tuned variants, this marks a substantial increase of 69.6% and 1,720%, respectively.

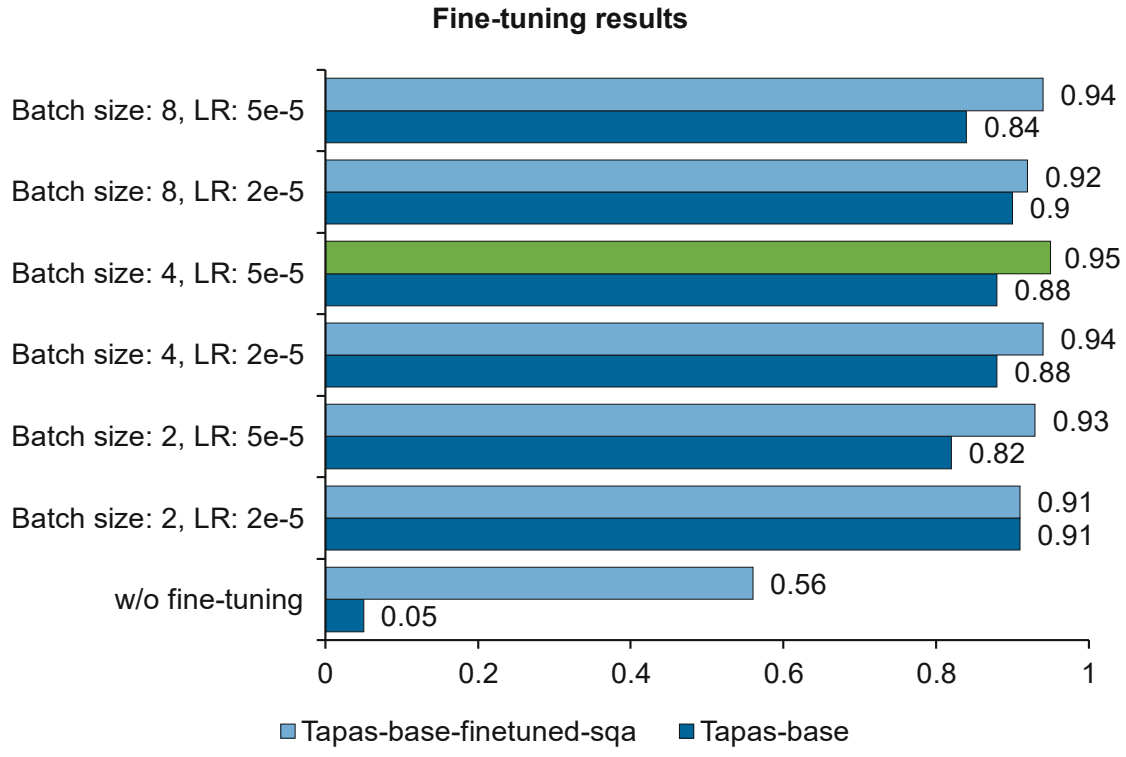


Figure 46 | Fine-tuning results, comparing the accuracy increase of both TaPas model checkpoints for all hyperparameter combinations

The respective training procedures are detailed in Figure 47, showing the training loss of the models for the training dataset across the fixed number of epochs. In the case of the optimal hyperparameter configuration, the training loss exhibits a consistent and gradual decline with each epoch. Conversely, alternative hyperparameter combinations demonstrate a slower reduction in training loss and spikes towards the end of the third epoch, potentially indicative of overfitting or a suboptimal training setup. Finally, these findings underscore the efficacy of leveraging the knowledge of the "Tapas-base-finetuned-sqa" model from fine-tuning on Microsoft's SQA dataset for increasing the domain-specific QA performance. These results make it the preferred choice over the base model for chatbot integration.

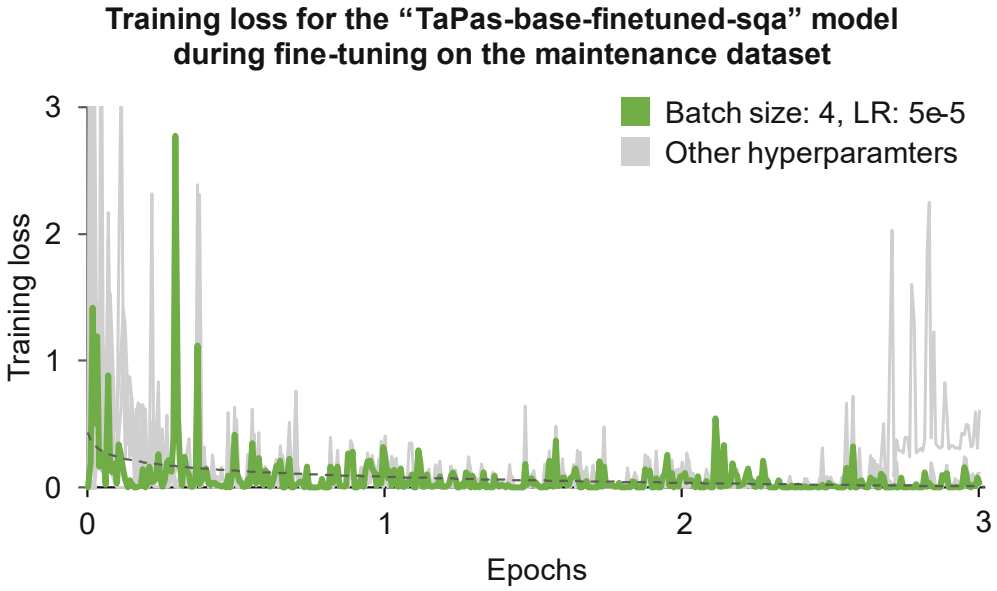


Figure 47 | The training loss shows that for optimal hyperparameter selection, the loss function continuously decreases over all three epochs

To evaluate the effects of the industrial maintenance-specific data augmentation approach on model performance, the best-performing model checkpoint, “Tapas-base-finetuned-sqa,” has been taken and fine-tuned on each augmented dataset separately, using the optimal hyperparameter combination from the prior step. The accuracy of the final models has been evaluated with the same exact match metric on the same test set as the non-augmented models. The results are summarized in Figure 48. Interestingly, it can be observed that all data augmentation methods lead to worse accuracy on the test set compared to the model trained on the non-augmented dataset. A significant adverse effect was injecting noise on a character level by simulating keyboard errors with an accuracy reduction of 15.79% and paraphrasing on a word level with a reduction of 9.57%, respectively. Figure 49 confirms these results by showing a significantly increased training loss at the end. Paraphrasing on a sentence level achieved comparable performance on the test set with a score of 0.91 and consistently decreased training loss across all epochs. It is essential to acknowledge that a decline in accuracy was anticipated due to using a relatively small dataset,

potentially leading to overfitting. Combined with an unchanged test set, the poor generalization capability and the discrepancy between training and test data lead to worse prediction results. However, the results underscore the potential of employing sentence-level paraphrasing to enhance diversity in table QA datasets within the context of industrial maintenance.

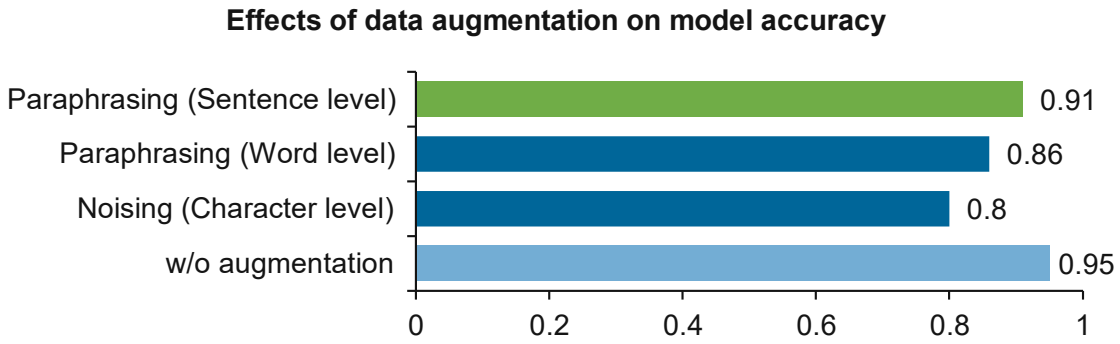


Figure 48 | Effects of data augmentation on fine-tuning results

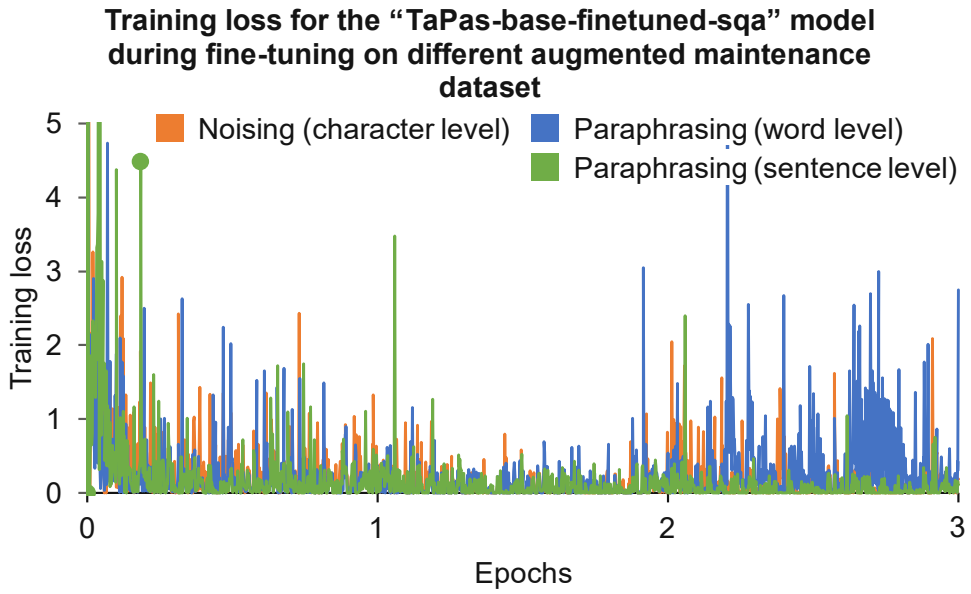


Figure 49 | Training loss curve for different augmented training sets.

In conclusion, the results and analysis above underscore the substantial improvement in LLM accuracy on QA downstream tasks when fine-tuned on industrial maintenance data. Additionally, the outcomes highlight the critical role of hyperparameter selection in the fine-tuning process, significantly influencing both the training procedure and final model performance. The comparison between models trained with augmented and non-augmented datasets underscores the necessity of domain-specific augmentation strategies to achieve a balance between increasing dataset diversity without distorting the original content and thereby negatively impacting the model's performance.

5.2 Maintenance Chatbot Usability

Industrial maintenance chatbots, designed to streamline communication and problem-solving processes, must be intuitive, efficient, and user-friendly to fulfill their potential to enhance productivity during maintenance activities and reduce downtime. In this context, the PSSUQ emerges as a crucial tool. Initially developed by researchers at Wichita University, PSSUQ is a well-established method for assessing users' satisfaction and perceived usability of software interfaces. Its structured approach provides valuable insights into the effectiveness, efficiency, and user satisfaction aspects of interactive systems.

5.2.1 User Interface

A comprehensive user interface is required to achieve an easy deployment of the LLM-based maintenance chatbot and smooth interaction with its users. This study adopted the chatbot widget made publicly available on GitHub by Gaikwad (2019). Figure 50 illustrates the user interface of the maintenance chatbot, showcasing segments of the maintenance dialogue outlined in Section 4.4.1.

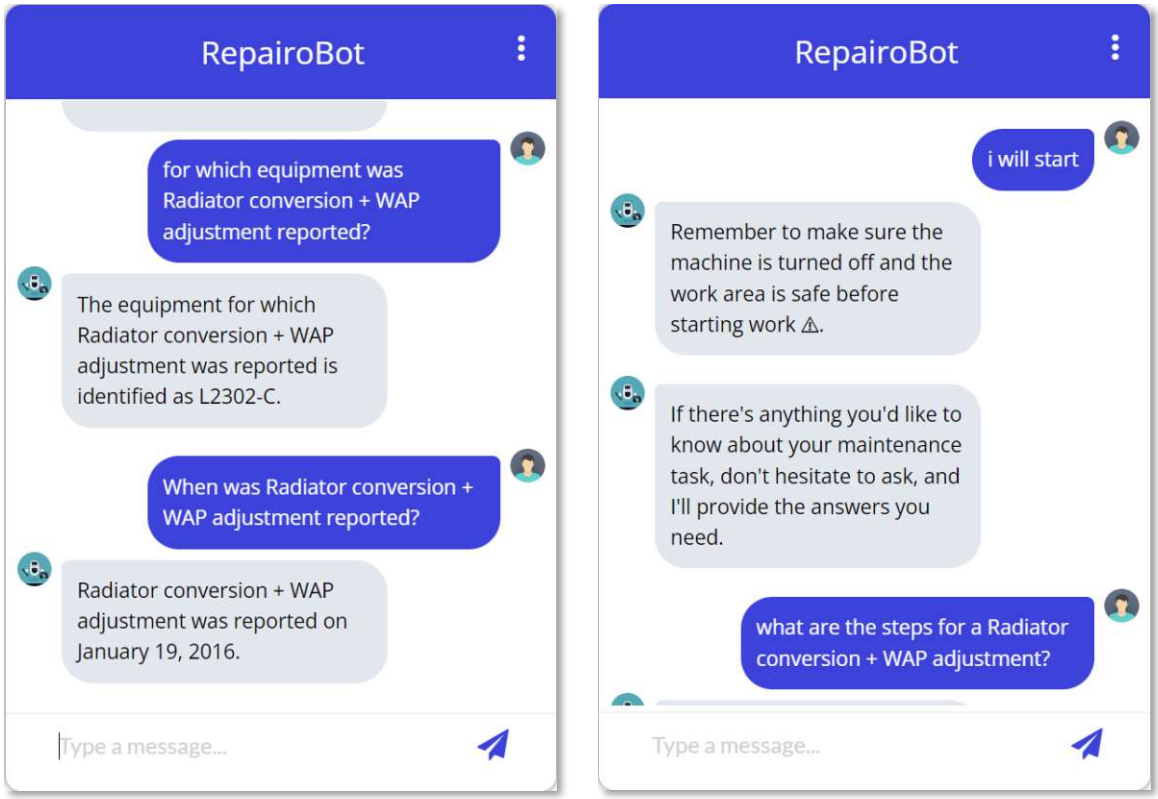


Figure 50 | The user interface offers maximum flexibility in deployment, functioning as an overlay widget. It includes multiple features, allowing the display of images, graphs, and interactive menus through clickable options.

Integration of this widget is straightforward, achieved through the Rasa server's rest channel defined in the credentials.yml file. It can be seamlessly embedded into any website or existing service within the industrial maintenance sector, enabling rapid

integration of the conversational agent into real-world operational scenarios. Further, its functionality and scope can be easily expanded, enhancing the user experience itself and maintenance operations through the integration of new features and data sources. This includes buttons, images, videos, PDF attachments, and charts, which can be incorporated to enrich the user interface.

5.2.2 Post-Study System Usability Questionnaire

This work evaluates the usability of the chatbot in real-world scenarios by utilizing the PSSUQ. Thus, this evaluation is not only a measure of the usability of the user interface but rather an evaluation of how users interact with the chatbot during maintenance tasks. It considers factors such as the chatbot's ability to understand user queries, the efficiency with which it provides relevant solutions, and the overall satisfaction users derive from the interaction. The data gathered through the PSSUQ will not only quantify user satisfaction but will also pinpoint specific areas where the chatbot interface excels and where it requires refinement.

The questionnaire utilizes a 7-point scale for responses to 16 questions, where a rating of 1 corresponds to "Strongly agree," and a rating of 7 corresponds to "Strongly disagree." Participants were also encouraged to provide qualitative comments after each question. The numerical ratings provide a quantitative basis for assessing the chatbot's usability, while the respective comments describe the context and reasoning behind the numerical responses. Together, the collected data provides a foundation for further discussions and analyses. The specific details regarding the questionnaire's overall structure and the individual questions are provided in Figures A. 2 and A. 3 in the appendix (J. R. Lewis, 1995).

The study conduction process of this work was designed to ensure objectivity, comparability, reproducibility, and quantifiable results and is separated into three stages:

1. **Software introduction:** Participants received a comprehensive introduction to the software, providing them with the necessary background knowledge and contextual understanding of the system.
2. **Software interaction:** Participants actively engaged with the software, navigating its features and functionalities.
3. **Software evaluation using PSSUQ:** Following the software interaction, participants evaluated the software's usability with the PSSUQ.

Overall, ten participants took part in the study. First, they were introduced to the chatbot system and the standardized 8-step maintenance process outlined in Section 4.4.1. Subsequently, participants had to interact with the software and were instructed to select one of the three down events suggested by the chatbot and attempt to follow the maintenance process, collecting necessary information about the event for its

proper execution. Upon successful completion, participants immediately proceeded to fill out the 16-step questionnaire, including their individual remarks.

5.2.3 User Satisfaction Analysis

The maintenance chatbot usability study results, presented in Figure 51, describe the overall usability score obtained from the PSSUQ and are categorized into three sub-domains:

- **System usefulness:** This category encompasses questions 1 to 6 of the questionnaire, focusing on how unfamiliar users can adapt to a new system and become proficient in its usage.
- **Information quality:** This category encompasses questions 7 to 12 and assesses the accuracy, relevance, and comprehensibility of the information provided by the chatbot.
- **Interface quality:** This category encompasses questions 13 to 16 and evaluates the design, layout, and navigational aspects of the chatbot interface, emphasizing user experience and interaction satisfaction.

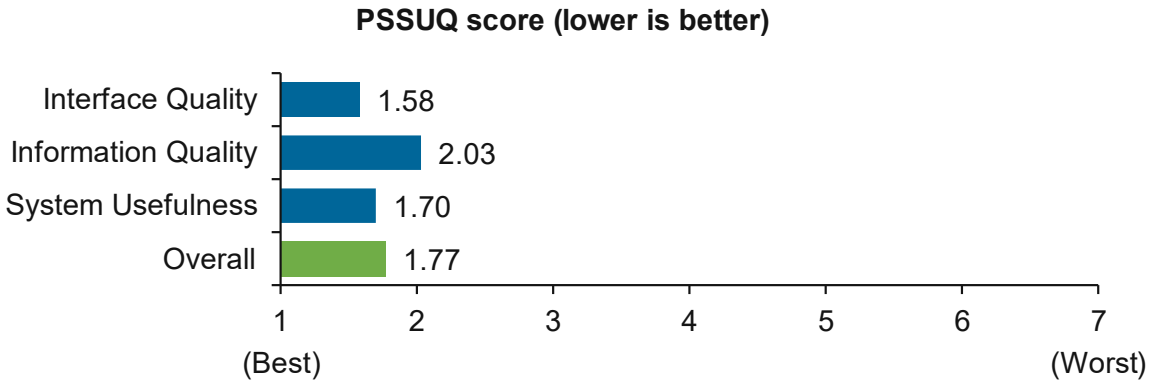


Figure 51 | LLM based maintenance chatbot usability scores determined using the PSSUQ

Notably, within the system usefulness category, the LLM-based industrial maintenance chatbot scored 1.77 points, nearing the highest attainable score. This performance underscores the advantages of LLM-driven chatbot systems. Unlike traditional chatbots reliant on predefined questions, LLM-based chatbots demonstrate increased adaptability, efficiently addressing a broad spectrum of user inquiries, both general and domain-specific. This is reflected by individual remarks by users, who, after receiving domain-specific instructions for a maintenance task, follow up with a general question that details the preceding answer. Thus, LLM-based maintenance chatbots offer enhanced user-friendliness by leveraging their extensive knowledge base derived from pre-training and fine-tuning, eliminating the need for users to learn specific utterances, questions, or input patterns comprehensible to the chatbot.

However, this flexibility comes at the cost of potentially less accurate information or answers provided to users, as evidenced by the information quality score of 2.03. While

most users acknowledge that the chatbot system supports them with accurate responses, the results are worse than observed in the previous category. This is summarized in a user's feedback to question 9: *"Depending on the formulation of the question, the information provided was clear/not clear."* An example of such uncertainty is presented in Figure 52, where the query "*What are the steps for a radiator conversion + WAP adjustment?"* yields an unclear response: *"What is the procedure for converting the radiator and adjusting the WAP?"* This lack of clarity arises from uncertainties and potential hallucinations within the GPT-3.5 model in the table question-answering pipeline. As depicted in the left section of Figure 52, although the correct answer was technically provided, the GPT-3.5 model failed to comprehend the task, resulting in an incorrect output.

Act as a maintenance chatbot and answer the following question, using the already provided answer. Answer in a complete sentence by paraphrasing the answer provided and including the question.

Answer: 1. HW ANko after chamber opening OK --> Anko required --> then Engineering . 2. chamber open. Checked WAP rings and gap. WAP conductance test with new setup parameters and 1.52mm WAP mechanism OK Wait LR Operator_1. . 3. WAP Conductance with Two-Piece Confinement Ring config not . 4. Original cooler installed, Wap mech. to 1.52 just. config changed, system pumps for .

Question: what are the steps for a Radiator conversion + WAP adjustment?

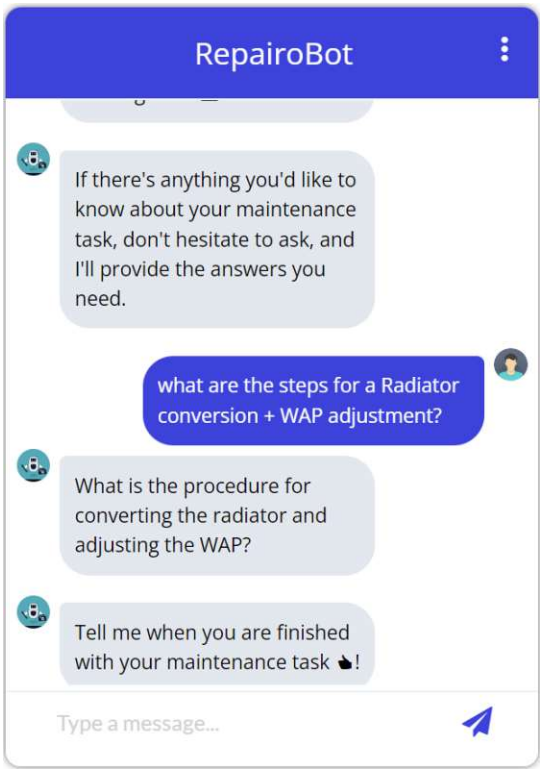


Figure 52 | Information quality of the maintenance chatbot suffers if GPT-3.5 fails to comprehend its task

The unclear response patterns further complicate the identification of mistakes or issues in the user input. This challenge is reflected in questions 7 (*"The system gave error messages that clearly told me how to fix problems."*) and 8 (*"Whenever I made a mistake using the system, I could recover easily and quickly."*), which received the lowest overall score of 3.11 on the PSSUQ. In the given example, the chatbot delivers an unclear response but continues along the predetermined conversational path, making it more challenging for the user to recover.

In the subcategory of interface quality, the maintenance chatbot received a score of 1.58, denoting general satisfaction with its user interface. Specific comments to the questionnaire's questions expressed that the information was presented in a structured

manner. However, users expressed a preference for supporting graphical materials during the maintenance operation.

In summary, the findings from the maintenance chatbot usability study, as assessed through the PSSUQ, indicate an overall positive user satisfaction with the system. The high score in the system usefulness subcategory underscores the easy adoption of the system for untrained users due to the integration of LLMs in the chatbot framework. Users appreciated the chatbot's ability to handle both domain-specific industrial maintenance queries and general inquiries related to unclear aspects of the maintenance process. However, participants expressed some concerns about the accuracy of instructions provided for maintenance tasks, highlighting the challenges associated with auto-regressive LLMs like GPT-3.5, which may sometimes produce unclear or distorted answers. A potential solution lies in enhancing the quality of data in the sub-table database, enabling better comprehension for models like GPT-3.5 and ensuring accurate outputs. Additionally, fine-tuning the prompt provided to the model or exploring further fine-tuning of auto-regressive models specifically for generative QA tasks in the industrial maintenance domain could address these issues.

6 Limitations

Several limitations should be considered in the context of this work and LLM-based industrial maintenance chatbots. Firstly, the literature review results concerning the best-suited LLMs for fine-tuning on domain-specific data are constrained due to the limited availability of the source code for most LLMs. Consequently, this is an issue for conducting public research, potentially resulting in gaps or underrepresenting recently published non-open-source LLMs in the literature review. Furthermore, while more recent publicly available LLMs may offer enhanced NLP performance compared to the models listed in this work (e.g., Llama 2), their large sizes, exceeding several gigabytes, would require high-performance computer hardware for practical fine-tuning, which was not available to the author.

Secondly, the absence of a real-world QA dataset required the creation of a semi-synthetic dataset using rule-based methods. Despite achieving good fine-tuning results, the lack of dataset diversity could lead to overfitting, potentially distorting the outcomes. Although this concern was mitigated by employing various data augmentation methods, the potential for a less robust system with reduced generalization capabilities remains. Notably, the models considered in this study were mainly pre-trained on English text data, resulting in significantly inferior performance when presented with non-English input data. This limitation could pose challenges in adapting the proposed approach to industries where non-English or multilingual documentation of maintenance processes is prevalent. It is also essential to note that the absence of real-world question-answering data resulted in eliminating sequential questions from the dataset, omitting the integration of SQA capabilities into the final maintenance chatbot. Additionally, while grid-search for optimal hyperparameters is a well-established scientific method, there is still a possibility of the optimization process getting stuck in a local minimum. To address this, expanding the search space for optimal hyperparameters through a grid extension or utilizing other hyperparameter optimization techniques should be explored, especially if more computational resources are available.

The table QA pipeline faces limitations due to the restricted encoding size of the TaPas model. Despite efforts to address this by dividing the maintenance log database into encodable sub-tables, there remains a risk of the model missing information distributed across multiple sub-tables during inference. One potential solution involves employing a domain-specific table-splitting strategy, separating the database based on equipment IDs or dates to enable the model to capture distributed information during the inference process.

The integration of auto-regressive LLMs into the question-answering pipeline, such as GPT-3.5, has a significant impact on the answer accuracy. The contextual understanding of user queries, especially in the domain-specific language of industrial

maintenance, can be challenging for LLMs, potentially resulting in misinterpretation and providing suboptimal responses. Additionally, evaluating answer accuracy for responses generated by GPT-3.5 is complex. Traditional exact match metrics are insufficient, as rephrased answers, while not identical to the answer cell, could still be correct. The F_1 score, as described in Section 5.1, offers a more suitable approach, quantifying the harmonic mean of precision and recall. Consequently, evaluation should eventually involve domain experts to ensure a comprehensive assessment.

Finally, in this context, it is imperative to consider compliance with forthcoming AI system regulations, like the EU AI Act, to develop responsible AI systems. An area of concern lies in the unclear data copyright situation associated with pre-training models like TaPas and GPT. Additionally, an examination of data governance measures, specifically focusing on data security and handling protocols implemented by third-party providers like OpenAI, is required. These uncertainties raise questions about the assurance that both providers adhere to future regulations, potentially risking their applicability within the EU market. However, definitive conclusions on this matter must await the publication of specific regulatory requirements.

7 Conclusion and Outlook

This thesis explores the application of LLMs in industrial maintenance contexts, aiming to assess the feasibility of LLM-based chatbots in this domain. It found that BERT-based models, GPT-2, and SentenceBERT are best suited for fine-tuning using domain-specific data, answering RQ1. These findings were made by systematically evaluating various LLMs found in scientific literature to determine their suitability for fine-tuning on domain-specific data. Despite the widespread popularity of OpenAI's GPT models, the study further reveals that BERT-based models emerge as the preferred choice for fine-tuning and subsequent integration into chatbot frameworks. This preference is attributed to their accessibility and versatility across various downstream tasks, especially when compared to larger auto-regressive models like GPT-3.

Second, the thesis highlights that LLMs can be effectively fine-tuned for table question-answering tasks using maintenance log tabular data, answering the first part of RQ2. It was demonstrated that a fine-tuned BERT-based TaPas model shows a substantial improvement in answer accuracy on the test dataset, achieving a nearly 70% increase compared to the non-fine-tuned counterpart. The study explores augmentation techniques to address the limitations posed by the limited dataset. It shows that a sentence-level paraphrasing augmentation method performs best on maintenance data because it increases data diversity while maintaining meaning and text semantics, answering RQ3. This thesis further identified the challenges to the successful application of data augmentation in the domain of industrial maintenance. In response, domain-specific data augmentation methods are proposed, including character and word-level noising and paraphrasing. The findings reveal that excessive distortion of original data can negatively impact fine-tuning results, underscoring the necessity for tailored domain-specific augmentation strategies.

The proposed maintenance chatbot demonstrates that fine-tuned LLMs can effectively be used as domain-specific question-answering components inside chatbot frameworks, answering the second part of RQ2. For this purpose, a table question-answering pipeline was developed using the fine-tuned TaPas model, which is integrated into a Rasa framework tailored to industrial maintenance applications. This LLM-driven maintenance chatbot follows a standardized 8-step maintenance process, providing support to users by answering general and domain-specific questions related to the maintenance task. The conducted usability study, assessed using the PSSUQ, highlighted significant advantages of integrating LLMs into chatbot frameworks within the context of industrial maintenance. Unlike conventional systems, the system does not require precise, predefined questions. Instead, it comprehends natural language queries, thus enhancing the user experience and accessibility of the system. Moreover,

the chatbot shows high versatility by addressing a wide range of general questions while also providing detailed, maintenance-specific answers.

Several areas for future work have been identified. Firstly, refining the TaPas model through fine-tuning with a maintenance-specific SQA dataset and integrating a sequential table question-answering pipeline into the chatbot framework would enable the chatbot to respond to domain-specific follow-up questions effectively. Additionally, collecting a real-world question-answering dataset specific to industrial maintenance would be a significant improvement to the existing dataset. Such a dataset, derived from authentic industrial scenarios, would improve the chatbot's knowledge base and enhance its comprehension of user queries. Furthermore, augmenting the chatbot responses with multimedia resources, such as videos and PDF documents, could provide users with comprehensive and detailed explanations, enhancing the chatbot's capability to actively support the industrial maintenance process.

In the context of maintenance management, the chatbot's scope could be extended to WP planning and WP execution (cf. Section 4.4.1) while being integrated into maintenance management systems. In addition to the assistance of workers during the manual execution of maintenance tasks, it could support the allocation of personnel, materials, and essential equipment and the determination of the task’s execution date.

Finally, providing a reliable LLM-based system necessitates particular attention to the elements of safety, transparency, and traceability in its output. Consequently, a comprehensive evaluation of compliance with existing and future regulatory frameworks becomes necessary, demanding a thorough analysis to guarantee the system's adherence to the high standards of ethical and regulatory considerations.

In summary, the LLM-based industrial maintenance chatbot, even when fine-tuned on limited maintenance data, demonstrates impressive capabilities. By addressing the identified areas for future work and limitations, LLM-based chatbots can effectively serve as supportive tools in industrial maintenance operations, significantly enhancing operational efficiency across various industries.

Bibliography

Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, *2*, 100006. https://doi.org/10.1016/j.mlwa.2020.100006

Alammar, J. (2018). *The Illustrated Transformer*. https://jalammar.github.io/illustrated-transformer/

Alammar, J. (2019). *The Illustrated GPT-2 (Visualizing Transformer Language Models)*. https://jalammar.github.io/illustrated-gpt2/

Ansari, F., & Glawar, R. (2022). Knowledge based Maintenance. In *Instandhaltungslogistik* (pp. 318–342). https://doi.org/10.3139/9783446470095

Ansari, F., Glawar, R., & Nemeth, T. (2019). PriMa: a prescriptive maintenance model for cyber-physical production systems. *International Journal of Computer Integrated Manufacturing*, *32*(4–5), 482–503. https://doi.org/10.1080/0951192X.2019.1571236

Ansari, F., Kohl, L., Giner, J., & Meier, H. (2021). Text mining for AI enhanced failure detection and availability optimization in production systems. *CIRP Annals*, *70*(1), 373–376. https://doi.org/10.1016/j.cirp.2021.04.045

Apt, W., Schubert, M., & Wischmann, S. (2018). *Digitale Assistenzsysteme Perspektiven und Herausforderungen für den Einsatz in Industrie und Dienstleistungen*. www.iit-berlin.de

Azunre, P. (2021). *Transfer Learning for Natural Language Processing*.

Bailan He. (2020). *Transfer Learning for NLP II | Modern Approaches in Natural Language Processing*. https://slds-lmu.github.io/seminar_nlp_ss20/transfer-learning-for-nlp-ii.html

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., … Liang, P. (2021). *On the Opportunities and Risks of Foundation Models*. http://arxiv.org/abs/2108.07258

British Standards Institution. (2018). *Maintenance - Maintenance terminology*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Openai, D. A. (2020). *Language Models are Few-Shot Learners*.

Brownlee, J. (2021). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Brownlee, J. (2022). *Difference Between a Batch and an Epoch in a Neural Network*. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Brügge, F. (2023). *Predictive maintenance market: 5 highlights for 2024 and beyond*. https://iot-analytics.com/predictive-maintenance-market/

Brundage, M. P., Sexton, T., Hodkiewicz, M., Dima, A., & Lukens, S. (2021). Technical language processing: Unlocking maintenance knowledge. *Manufacturing Letters*, *27*, 42–46. https://doi.org/10.1016/J.MFGLET.2020.11.001

Cervo, H., Verpoorten, J., Farioli, S., Morachioli, S., & Gluchoswski, M. (2023). *Maintenance and operations: Is asset productivity broken?*

Chakraborty, D., Ghosh, A., & Saha, S. (2019). A survey on internet-of-thing applications using electroencephalogram. In *Emergence of Pharmaceutical Industry Growth with Industrial IoT Approach* (pp. 21–47). Elsevier. https://doi.org/10.1016/B978-0-12-819593-2.00002-9

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., … Fiedel, N. (2022). *PaLM: Scaling Language Modeling with Pathways*. https://arxiv.org/abs/2204.02311v5

Chui, M., Roberts, R., & Yee, L. (2022). *Generative AI is here: How tools like ChatGPT could change your business*.

Coli, E., Melluso, N., Fantoni, G., & Mazzei, D. (2020). *Towards Automatic building of Human-Machine Conversational System to support Maintenance Processes*. https://doi.org/10.48550/arxiv.2005.06517

Deighton, M. (2016). *Facility integrity management : effective principles and practices for the oil, gas and petrochemical industries*.

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 4171–4186. https://github.com/tensorflow/tensor2tensor

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. http://arxiv.org/abs/1810.04805

Dima, A., Lukens, S., Hodkiewicz, M., Sexton, T., & Brundage, M. P. (2021). Adapting natural language processing for technical text. *Applied AI Letters*, *2*(3). https://doi.org/10.1002/AIL2.33

ERPS. (2023). *BRIEFING EU Artificial intelligence act*.

Feigenblat, G., Gunasekara, C., Sznajder, B., Aaronov, R., Konopnicki, D., & Joshi, S. (2021). TWEETSUMM - A Dialog Summarization Dataset for Customer Service. *Findings of the Association for Computational Linguistics, Findings of ACL: EMNLP 2021*, 245–260. https://doi.org/10.18653/V1/2021.FINDINGS-EMNLP.24

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., & Hovy, E. (2021). *A Survey of Data Augmentation Approaches for NLP*. https://github.com/styfeng/DataAug4NLP.

Firsanova, V. (2021). *Supporting the Inclusion of People with Asperger Syndrome: Building a Customizable Chatbot with Transfer Learning*. https://esignals.fi/research/en/2021/01/20/supporting

Flare. (2021). *Difference Between Natural & Technical Language Processing*. https://www.flare-solutions.com/news/the-difference-between-natural-language-processing-and-technical-language-processing/

Gaikwad, J. (2019). *GitHub - JiteshGaikwad/Chatbot-Widget*. https://github.com/JiteshGaikwad/Chatbot-Widget/tree/main

Gholami, S., & Omar, M. (2023). *Does Synthetic Data Make Large Language Models More Efficient?*

Glawar, R., Ansari, F., Reichsthaler, L., Sihn, W., & Toth, D. (2022). Maintenance-Free Factory: A Holistic Approach for Enabling Sustainable Production Management. *IFAC-PapersOnLine*, *55*(10), 2318–2323. https://doi.org/10.1016/J.IFACOL.2022.10.054

Glawar, R., Nemeth, T., & Kovacs, K. (2022). Digitale Transformation in der Instandhaltung. In *Instandhaltungslogistik* (pp. 283–333). Carl Hanser Verlag GmbH & Co. KG. https://doi.org/10.3139/9783446470095.012

Hagiwara, M. (2021). *Real-World Natural Language Processing*. Manning Publications Co.

Herzig, J., Nowak, P. K., Uller, T. M. ¨, Piccinno, F., & Eisenschlos, J. M. (2020). *TAPAS: Weakly Supervised Table Parsing via Pre-training*.

Hock-Chuan, C. (2018). *Regular Expression (Regex) Tutorial*. https://www3.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. de Las, Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., Driessche, G. van den, Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., … Sifre, L. (2022). *Training Compute-Optimal Large Language Models*. https://doi.org/10.48550/arxiv.2203.15556

Hugging Face. (2023a). *Encoder models*. https://huggingface.co/learn/nlp-course/chapter1/5?fw=pt

Hugging Face. (2023b). *Exact Match*. https://huggingface.co/spaces/evaluate-metric/exact_match

Hugging Face. (2023c). *Sequence-to-sequence models*. https://huggingface.co/learn/nlp-course/chapter1/7?fw=pt

Hugging Face. (2023d). *What is Question Answering?* https://huggingface.co/tasks/question-answering

Iyyer, M., Yih, W.-T., & Chang, M.-W. (2016). *Answering Complicated Question Intents Expressed in Decomposed Question Sequences*. http://aka.ms/sqa

Janiesch, C., Zschech, P., & Heinrich, K. (2021). *Machine learning and deep learning*. https://doi.org/10.1007/s12525-021-00475-2

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Heafield, H. H. K., Neckermann, T., Seide, F., Germann, U., Aji, A. F., Bogoychev, N., Martins, A. F. T., & Birch, A. (2018). Marian: Fast Neural Machine Translation in C++. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of System Demonstrations*, 116–121. https://doi.org/10.18653/v1/p18-4020

Jurafsky, D., & James H, M. (2023). *Speech and Language Processing A.1 Markov Chains*.

Kapoor, A., & Shetty, S. (2023). *Implementing Natural Language Generation Through Industry-Specific Chatbots*. https://doi.org/10.56155/978-81-955020-2-8-6

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, *82*(3), 3713–3744. https://doi.org/10.1007/s11042-022-13428-4

Knote, R., Janson, A., Söllner, M., & Leimeister, J. M. (2019). *Classifying Smart Personal Assistants: An Empirical Cluster Analysis*. https://hdl.handle.net/10125/59642

Kochmar, E. (2022). *Getting Started with Natural Language Processing*.

Krishnan, J., Coronado, P., Purohit, H., & Rangwala, H. (2020). Common-Knowledge Concept Recognition for SEVA. *CEUR Workshop Proceedings*, *2600*. https://arxiv.org/abs/2003.11687v1

Lewis, J. R. (1995). IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, *7*(1), 57–78. https://doi.org/10.1080/10447319509526110

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*.

Li, B., Hou, Y., & Che, W. (2022). Data augmentation approaches in natural language processing: A survey. *AI Open*, *3*, 71–90. https://doi.org/10.1016/J.AIOPEN.2022.03.001

Li, Y., Li, Z., Zhang, K., Dan, R., & Zhang, Y. (2023). *ChatDoctor: A Medical Chat Model Fine-tuned on LLaMA Model using Medical Domain Knowledge*. https://github.com/Kent0n-Li/ChatDoctor.

Lin, T.-H., Huang, Y.-H., & Putranto, A. (2022). *Intelligent question and answer system for building information modeling and artificial intelligence of things based on the bidirectional encoder representations from transformers model*. https://doi.org/10.1016/j.autcon.2022.104483

Lipenkova, J. (2022). *Choosing the right language model for your NLP use case*. Towardsdatascience. https://towardsdatascience.com/choosing-the-right-language-model-for-your-nlp-use-case-1288ef3c4929

Ma, E. (2019). *NLP Augmentation*. Https://Github.Com/Makcedward/Nlpaug.

Matyas, K. (2022a). Instandhaltung. In *Instandhaltungslogistik* (pp. 27–62). Carl Hanser Verlag GmbH & Co. KG. https://doi.org/10.3139/9783446470095

Matyas, K. (2022b). Lean Maintenance. In *Instandhaltungslogistik* (pp. 183–203). Carl Hanser Verlag GmbH & Co. KG. https://doi.org/10.3139/9783446470095

Mehboob, F. A., Mahmood Malik, K., Khader Jilani Saudagar, A., Rauf Knightec, A. A., Jiang, R., Badruddin Khan, M., AlTameem, A., Mehboob, F., & Rauf, A. (2023). *Medical Report Generation and Chatbot for COVID_19 Diagnosis Using Open-AI*. https://doi.org/10.21203/RS.3.RS-2563448/V1

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1993). *Introduction to WordNet: An On-line Lexical Database*.

Minutolo, A., Damiano, E., De Pietro, G., Fujita, H., & Esposito, M. (2022). A conversational agent for querying Italian Patient Information Leaflets and

improving health literacy. *Computers in Biology and Medicine*, *141*, 105004. https://doi.org/10.1016/j.compbiomed.2021.105004

Montenegro, J. L. Z., & da Costa, C. A. (2022). The HoPE Model Architecture: a Novel Approach to Pregnancy Information Retrieval Based on Conversational Agents. *Journal of Healthcare Informatics Research*, *6*(3), 253–294. https://doi.org/10.1007/S41666-022-00115-0/FIGURES/16

Müller, T. (2020). *Using Neural Networks to Find Answers in Tables*. https://blog.research.google/2020/04/using-neural-networks-to-find-answers.html

Nair, R. (2023). *Industry 4.0 check-in: 5 learnings from ongoing digital transformation initiatives*. https://iot-analytics.com/industry-4-0-check-in-5-learnings-from-digital-transformation-initiatives/

Nowakowski, T., Tubis, A., & Werbińska-Wojciechowska, S. (2019). Evolution of technical systems maintenance approaches – Review and a case study. *Advances in Intelligent Systems and Computing*, *835*, 161–174. https://doi.org/10.1007/978-3-319-97490-3_16/TABLES/3

Obadinma, S., Khan Khattak, F., Wang, S., Sidhom, T., Lau, E., Robertson, S., Niu, J., Au, W., Munim, A., Raja, K., Bhaskar, K., Wei, B., Ren, I., Muhammad, W., Li, E., Ishola, B., Wang, M., Tanner, G., Shiah, Y.-J., … Dolatabadi, E. (2023). *Bringing the State-of-the-Art to Customers: A Neural Agent Assistant Framework for Customer Service Support*. https://arxiv.org/abs/2302.03222v1

OpenAI. (2023). *GPT-4 Technical Report*.

Page, M. J., Mckenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., Mcdonald, S., … Moher, D. (2020). *The PRISMA 2020 statement: an updated guideline for reporting systematic reviews*. https://doi.org/10.1136/bmj.n71

Pan, F., Canim, M., Glass, M., Gliozzo, A., & Fox, P. (2021). CLTR: An End-to-End, Transformer-Based System for Cell Level Table Retrieval and Table Question Answering. *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the System Demonstrations*, 202–209. https://doi.org/10.18653/v1/2021.acl-demo.24

Pasupat, P., & Liang, P. (2015). Compositional Semantic Parsing on Semi-Structured Tables. *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural*

Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference, *1*, 1470–1480. https://doi.org/10.3115/v1/p15-1142

Peffers, J. (2006). *The Design Science Research Process : A Model for Producing and Presenting Information Systems Research*. 83–106. http://rightsstatements.org/page/InC/1.0/?language=en

Pereira, R., Lima, C., Pinto, T., & Reis, A. (2023). Virtual Assistants in Industry 4.0: A Systematic Literature Review. *Electronics 2023, Vol. 12, Page 4096*, *12*(19), 4096. https://doi.org/10.3390/ELECTRONICS12194096

Peres, R. S., Jia, X., Lee, J., Sun, K., Colombo, A. W., & Barata, J. (2020). Industrial Artificial Intelligence in Industry 4.0 -Systematic Review, Challenges and Outlook. *IEEE Access*. https://doi.org/10.1109/ACCESS.2020.3042874

Peters, M., Ruder, S., & Smith, N. A. (2019). *To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks*.

Raaijmakers, S. (2020). *Deep Learning for Natural Language Processing MEAP V07*. www.manning.com/

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, *21*, 1–67. http://jmlr.org/papers/v21/20-074.html.

Rajpurkar, P., Jia, R., & Liang, P. (2018). *Know What You Don't Know: Unanswerable Questions for SQuAD*.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. https://stanford-qa.com,

Rasa Technologies. (2023a). *Introduction to Rasa Open Source & Rasa Pro*. https://rasa.com/docs/rasa/

Rasa Technologies. (2023b). *Rasa Architecture Overview*. https://rasa.com/docs/rasa/arch-overview/

Rastogi, R. (2023). *ML-Papers-Explained: Explanation to key concepts in ML*. https://github.com/dair-ai/ML-Papers-Explained

Reimers, N. (2022). *Pretrained Models — Sentence-Transformers documentation*. https://www.sbert.net/docs/pretrained_models.html

Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. https://github.com/UKPLab/

Rogge, N. (2023). *Fine-tuning TapasForQuestionAnswering on SQA*. https://colab.research.google.com/github/NielsRogge/Transformers-Tutorials/blob/master/TAPAS/Fine_tuning_TapasForQuestionAnswering_on_SQA.ipynb#scrollTo=l5Ds1ZM41KC9

Roth, A. (2016). Einführung und Umsetzung von Industrie 4.0. In *Einführung und Umsetzung von Industrie 4.0*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-48505-7

Rothman, D. (2021). *Transformers for Natural Language Processing*. www.packt.com

Schnelzer, J. (2019). *Differentiation of Industry 4.0 Models. The 4th Industrial Revolution from different Regional Perspectives in the Global North and Global South*. https://doi.org/10.13140/RG.2.2.35510.55363

Silvestri, L., Forcina, A., Introna, V., Santolamazza, A., & Cesarotti, V. (2020). Maintenance transformation through Industry 4.0 technologies: A systematic literature review. *Computers in Industry*, *123*, 103335. https://doi.org/10.1016/J.COMPIND.2020.103335

Smith, S. L., Kindermans, P. J., Ying, C., & Le, Q. V. (2017). Don't Decay the Learning Rate, Increase the Batch Size. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. https://arxiv.org/abs/1711.00489v2

Stanford NLP Group. (2013). *Stanford TokensRegex*. https://nlp.stanford.edu/software/tokensregex.html#Usage

Thomas, D. S., & Weiss, B. A. (2020). *Economics of Manufacturing Machinery Maintenance A Survey and Analysis of U.S. Costs and Benefits NIST Advanced Manufacturing Series 100-34*. https://doi.org/10.6028/NIST.AMS.100-34

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., … Scialom, T. (2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models*.

Tunstall, L., Von Werra, L., & Wolf, T. (2022). *Natural Language Processing with Transformers Building Language Applications with Hugging Face*.

Vajjala, S., Majumder, B., Gupta, A., & Surana, H. (2020). *Practical Natural Language Processing A Comprehensive Guide to Building Real-World NLP Systems*.

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need*.

Vishwanathan, A., Warrier, R. U., Suresh, G. V., & Kandpal, C. S. (2023). *Multi-Tenant Optimization For Few-Shot Task-Oriented FAQ Retrieval*. https://arxiv.org/abs/2301.10517v1

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). *GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTAND-ING*.

Wang, L., Mujib, M. I., Williams, J., Demiris, G., & Huh-Yoo, J. (2021). *An Evaluation of Generative Pre-Training Model-based Therapy Chatbot for Caregivers*. https://arxiv.org/abs/2107.13115v1

Wellsandt, S., Hribernik, K., & Thoben, K. D. (2021). Anatomy of a Digital Assistant. *IFIP Advances in Information and Communication Technology*, *633 IFIP*, 321–330. https://doi.org/10.1007/978-3-030-85910-7_34

Yan, X., & Nakashole, N. (2021). *A Grounded Well-being Conversational Agent with Multiple Interaction Modes: Preliminary Results*. 143–151. https://doi.org/10.18653/v1/2021.nlp4posimpact-1.16

Yih, W.-T., He, X., & Meek, C. (2014). *Semantic Parsing for Single-Relation Question Answering*. 643–648.

Yıldırım, S., & Asgari-Chenaghlu, M. (2021). *Mastering transformers : build SOTA models from scratch with advanced natural language processing techniques*.

Zeng, G., Yang, W., Ju, Z., Yang, Y., Wang, S., Zhang, R., Zhou, M., Zeng, J., Dong, X., Zhang, R., Fang, H., Zhu, P., Chen, S., & Xie, P. (2020). MedDialog: Large-scale Medical Dialogue Datasets. *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 9241–9250. https://doi.org/10.18653/V1/2020.EMNLP-MAIN.743

Zhong, V., Xiong, C., & Socher, R. (2017). *Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning*. https://arxiv.org/abs/1709.00103v7

Zonta, T., da Costa, C. A., da Rosa Righi, R., de Lima, M. J., da Trindade, E. S., & Li, G. P. (2020). Predictive maintenance in the Industry 4.0: A systematic literature review. *Computers and Industrial Engineering*, *150*. https://doi.org/10.1016/j.cie.2020.106889

Table of Figures

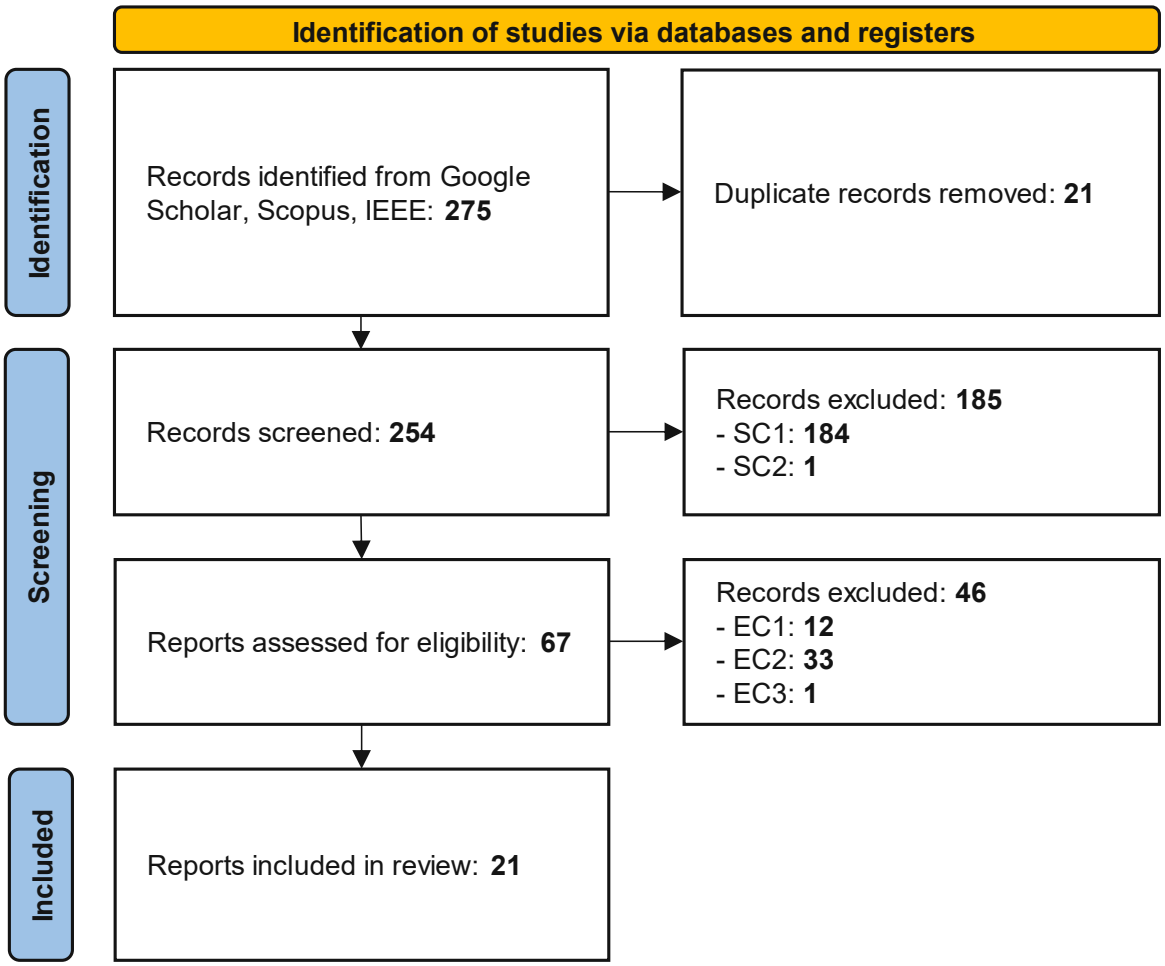
Table of Tables

Table of Abbreviations

AI	Artificial Intelligence
AR	Augmented Reality
BERT	Bi-directional Encoder Representations from Transformers
cf.	Confer
DL	Deep Learning
DSRP	Design Science Research Process
e.g.	Exempli Gratia
FAQ	Frequently Asked Question
GDPR	General Data Protection Regulation
HMM	Hidden Markov models
IoT	Internet of Things
IS	Information Systems
KBM	Knowledge Based Maintenance
KI	Künstliche Intelligenz
LLM	Large Language Model
LSTM	Long Short Term Memory
M2F	Maintenance-Free Factory
ML	Machine Learning
MTTR	Mean Time to Repair
NER	Named Entity Recognition
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
O	Objective
P	Problem
PriMa	Prescriptive Maintenance Model
PSSUQ	Post-Study Usability Questionnaire
QA	Question Answering
RNN	Recurrent Neural Networks
RQ	Research Question
SDK	Software Development Kit
SQA	Sequential Question Answering
STS	Semantic Text Similarity
SVM	Support Vector Machine
TC	Text Classification
TG	Text Generation
TLP	Technical Language Processing

TPM	Total Productive Maintenance
WP	Work preparation

Appendix



A. 1 | Identification report as provided by the PRISMA statement

PSSUQ	1	2	3	4	5	6	7	N.A.
1. Overall, I am satisfied with how easy it is to use this system.								
2. It was simple to use this system.								
3. I was able to complete the tasks and scenarios quickly using this system.								
4. I felt comfortable using this system.								
5. It was easy to learn to use this system.								
6. I believe I could become productive quickly using this system.								
7. The system gave error messages that clearly told me how to fix problems.								
8. Whenever I made a mistake using the system, I could recover easily and quickly.								
9. The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.								

A. 2 | PPSSQ questions 1-9

	1	2	3	4	5	6	7	N.A.
10. It was easy to find the information I needed.								
11. The information was effective in helping me complete the tasks and scenarios.								
12. The organization of information on the system screens was clear.								
13. The interface of this system was pleasant.								
14. I liked using the interface of this system.								
15. This system has all the functions and capabilities I expect it to have.								
16. Overall, I am satisfied with this system.								

A. 3 | PPSSQ questions 10-16