Daniel Koch · Andreas Kohne ·
Nils Brechbühler

Prompt Engineering in the Enterprise – An Introduction

Competitive Advantages through Generative AI and Large Language Models

Springer

Prompt Engineering in the Enterprise—An Introduction

Daniel Koch · Andreas Kohne ·
Nils Brechbühler

Prompt Engineering in the Enterprise—An Introduction

Competitive Advantages through Generative AI and Large Language Models

Springer

Daniel Koch
Auetal, Niedersachsen
Germany

Andreas Kohne
Hessisch Oldendorf, Germany

Nils Brechbühler
Barsinghausen, Germany

ISBN 978-3-658-49217-5 ISBN 978-3-658-49218-2 (eBook)
https://doi.org/10.1007/978-3-658-49218-2

This Springer imprint is published by the registered company Springer Fachmedien Wiesbaden GmbH, part of Springer Nature.
The registered company address is: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Foreword by Prof. Dr. Knut Linke

In an increasingly digitalized and self-organizing work environment, the field of Artificial Intelligence (AI) plays a central role. The rapid development of technologies such as Generative Pre-trained Transformers (GPTs) and similar assistants has the potential to revolutionize processes and the way work is organized and performed. In this context, the concept of "New Work"—the transformation of traditional work models toward more flexible, innovative, and human-centered structures—continues to gain significance.

In this book, the authors provide you with a fundamental, well-founded, and systematic introduction to the field of "Prompt Engineering"—a tool that is central to the effective use of AI. The authors examine the essential and diverse aspects of prompt engineering and the related area of generative AI.

In addition to introducing how AI is establishing itself as a disruptive technology, this section presents and explains the fundamentals of AI. This is particularly important, as the terminology explained here can be considered general knowledge in the field of AI and its context. Understanding these basics also makes it easier to use AI systems such as ChatGPT, Claude, or Microsoft Copilot. As a reader, you will gain an

overview of the fundamentals of AI models, especially GPTs, and their technical details, such as tokens and context windows. This provides the necessary foundation for a deep understanding of the subsequent content of the book.

Following the fundamentals of prompt engineering presented in the second chapter, which explains the concepts underlying the prompting approach and introduces initial forms of interaction with AI systems, advanced approaches are introduced step by step. This foundational understanding of prompts and prompting is essential to ensure that AI systems can be properly and successfully utilized. It is important to note that the knowledge conveyed in this book is uniformly applicable to all generative AI systems.

Arising from this necessity for effective and successful management of AI systems through well-considered prompting, various prompting frameworks will be introduced and applied in the following sections. These frameworks, such as the R-T-F, T-A-G, B-A-B, C-A-R-E, and R-I-S-E frameworks, offer structured approaches to systematize and optimize the interaction between humans and AI systems.

Especially in the context of New Work, where individuals must organize themselves to a high degree and make independent decisions, the use of AI systems for reflection and exchange will continue to increase significantly. Various prompting frameworks are particularly well suited to achieving above-average results in this regard. Their clear structure and detailed descriptions also make it possible to apply these frameworks directly in the workplace, thereby enhancing the quality of work.

The techniques presented later in this textbook, such as zero-, one-, and few-shot prompts as well as chain-of-thought prompts, offer advanced approaches for optimizing the use of AI systems. They provide insight into the future of AI systems. Emerging thinking AI systems with thinking tags currently employ similar approaches to the methods introduced here.

By mastering these advanced methods, companies and their employees can gain a better understanding of the processes within AI systems and, hopefully, develop new and innovative processes and products that meet the demands of the world.

Prof. Dr. Knut Linke
Professor of Computer Science at the International University (IU)

In addition to his role as a professor in the dual study program at IU, Prof. Dr. Linke supervises and designs various German- and English-language bachelor's and master's courses in the field of AI prompting within IU's distance learning offerings.

Preface

Artificial intelligence (AI) has been a widely discussed topic since the launch of ChatGPT by OpenAI at the end of 2022. In particular, generative AI—systems that create text, images, videos, or audio (such as speech or music)—has led to a global breakthrough in both awareness and application.

In particular, large language models (LLMs) are now seeing widespread use. And this is happening everywhere: from industry and public administration to the private sector. AI is not a local phenomenon or something that affects only specific industries or countries. AI now impacts everything and everyone.

Despite the pervasive media coverage, the topics of AI—and especially the opportunities and possibilities for optimizing business and administrative processes—have still not reached all areas. There are various reasons for this. One of the main reasons is the lack of expertise in industry and public administration.

We now recommend that everyone engage with this current and forward-looking topic. To illustrate this, we like to use a simple metaphor:

Every young child learns very early in life that, as a rule, there is a light switch on the wall to the left or right of a door. Press the

switch—and the light turns on. In this simple way, we make use of a technology that many people, even as adults, do not fully understand: What exactly is electricity? Where does it come from? Where does it go? And what are the physical principles and laws behind it? I do not need to understand all of this. Nevertheless, I can make use of the technology with a basic understanding. Many people are also able to install a new lamp or replace a light bulb on their own. In addition to the benefits, it is hopefully also conveyed early in a child's life that there are dangers as well. For example, no one should touch exposed wires that may be sticking out of a wall in a building under construction, and nothing—especially nothing conductive—or even fingers should be inserted into an electrical socket, as this poses an immediate danger. Artificial intelligence should be understood by everyone at this level in the near future, as the technology is spreading and evolving at a rapid pace. To use it successfully, I do not need to understand at the deepest mathematical level how the systems work. Nevertheless, I can take advantage of their benefits.

In many companies, the transformative power of AI technology is unfortunately still not fully understood. This is partly because simply "using" the new technology sometimes leads to subpar results that fall short of expectations. Many of our clients report that they tried to solve a task using ChatGPT and similar tools, but were disappointed with the outcome. This often leads to the decision to dismiss the technology, at least for the time being, as not yet mature and to continue with "business as usual." In our experience, the main problem in such cases is that users fall into a typical AI trap: they assume that, since AI systems (especially LLMs) can be operated "easily" using natural language, no further guidance or training is required. Unfortunately, this is not the case. While the systems do provide answers to the questions asked (the so-called prompts), these responses are often flawed, ambiguous, insufficiently specific, or simply incorrect. This understandably leads to frustration and confusion among users.

The solution to this problem is called prompt engineering. This refers to techniques that structure queries to AI-based systems in such a way that they deliver optimal results. In many cases, a poor prompt will yield poor results even with a high-quality LLM, whereas a well-crafted prompt can produce excellent results even with a less advanced LLM.

By using techniques from prompt engineering, it is possible to create highly effective queries for AI systems, which in turn deliver the desired quality of responses.

We regard prompt engineering as an important "future skill" that enables individuals to navigate the world of AI systems confidently, both now and in the future.

In this book, we have compiled our experiences from projects, workshops, and training sessions with clients in industry, public administration, and the broadcasting sector over recent years, presenting them in a way that makes the various concepts of working with AI-based systems easy to understand and immediately applicable in everyday practice.

With this book, we aim to encourage active engagement with the technology and to seize the opportunities and possibilities it offers for businesses.

True to our motto: "Together—courageous—vigilant," we wish you every success on your journey toward effective AI utilization.

January 2025
Hanover

Daniel Koch
Dr. Andreas
Nils Brechbühler
Managing Director of Valisory GmbH
Responsible for getting things done!

Note For reasons of readabilitsy, the text in this book uses the generic masculine form. However, we explicitly address all people and genders equally.

Contents

About the Authors

Daniel Koch As a serial entrepreneur and expert in digitalization and agile transformation, Daniel Koch has been combining in-depth technological expertise with business acumen for over a decade. His publications on business and technology provide practical insights into the digital transformation of organizations. With a background in software development and as a certified Agile Coach, he excels at successfully

integrating technological innovations into business processes. His academic credentials as a lecturer at Hochschule Weserbergland and Berufliche Hochschule Hamburg, as well as his role as a member of the university senate at HSW, underscore his competence in both theory and practice. Daniel Koch holds a B.Sc. in Business Informatics and an MBA in General Management, leveraging this interdisciplinary education to successfully shape complex digital transformation processes. As Managing Director of Valisory GmbH, he is responsible for the strategic and operational implementation of marketing and sales.

Valisory GmbH enables medium-sized companies to become AI-ready within just a few days, supporting them from initial insights through to successful implementation. Through practice-oriented workshops and hands-on implementation support, Valisory's clients achieve measurable results quickly, without getting lost in technical details.

Dr. Andreas Kohne As an expert in innovation, transformation, and communication, Andreas Kohne publishes relevant expertise in a concise and accessible manner. His publications are considered standard reading in business and academia both nationally and internationally, and are available in German and English.

The author and editor is an experienced management consultant who supports his clients on their path to a successful and relevant future. He is the founder and managing director of Valisory GmbH and itkon GmbH. As a sought-after speaker, Andreas Kohne imparts practical, industry-relevant expertise. With a compelling blend of expertise, interaction, and motivation, he works nationally and internationally as a tech translator. He succeeds in making complex digital structures and processes accessible and easy to understand. Currently, he advises

clients in industry, public administration, and broadcasting on the forward-looking, profitable, and secure implementation of artificial intelligence.

Additionally, he works as a lecturer in the fields of business administration and computer science at Hochschule Fresenius and Hochschule Weserbergland. He is a multiple award-winning TOP EXPERT of the BVMID (Federal Association of Small and Medium-Sized Enterprises in Germany).

Andreas Kohne is an appointed senator in the European Economic Senate.

Nils Brechbühler Nils Brechbühler is an experienced practitioner and leading expert in digital transformation and the modernization of public administration. As founder and managing director of Valisory GmbH and Brechbühler GmbH, he supports public institutions, companies, and public broadcasters in the successful implementation of digitization and transformation projects.

With over 15 years of experience in the public sector, complemented by leadership roles at IT service providers and internationally active consulting firms, he possesses a deep understanding of the specific challenges and needs of government, business, and media. In more than 1000 workshops and over 500 projects, he has demonstrated how complex topics such as public sector modernization, digital transformation,

leadership, and communication can be translated into tangible and sustainable solutions.

Nils Brechbühler is not only a member and speaker at the National E-Government Competence Center (NEGZ), but also a sought-after lecturer at universities, professional conferences, and working groups at all federal levels. He is currently focusing intensively on the opportunities and challenges of using artificial intelligence in public administration, with the aim of integrating these technologies into transformation processes in a practical and responsible manner.

1

Introduction

Abstract This chapter first explains the impact of artificial intelligence (AI) on businesses. Potential use cases as well as the limitations of the technology are presented. The chapter then describes how companies can gradually approach the adoption of this technology. Finally, the effects of AI on employee roles and skills are discussed.

AI (Artificial Intelligence) appears to have become an omnipresent topic for companies. Understandably so, as its capabilities are often impressive, even breathtaking. With the right tools, companies can work faster and more efficiently, reduce costs, relieve employees, increase transparency within the organization—and much more.

In particular, the field of generative AI is increasingly making its way into the daily operations of many companies. To leverage the benefits of these systems, organizations must engage with the technology and deploy it in a meaningful and effective manner. A crucial aspect of this is prompt engineering, which addresses the question of how to formulate queries to such systems in order to achieve the best possible results.

This book explains how AI works in general, with a particular focus on generative language models, and outlines their relevance and

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_1

potential for businesses. It demonstrates how to craft effective prompts for such language models, introducing various proven structures and techniques. The book also explores how prompts can be generated using language models themselves (metaprompting). Finally, it discusses the application of prompt engineering in configuring autonomous AI agents.

1.1 AI—a Disruptive Technology for Business?

Since around the end of 2022, it seems that the technology discourse has revolved around a single topic: AI. Whether in society, politics, public administration, or business—AI appears to be everywhere. AI is experiencing a hype similar to that of other innovations (such as blockchain technology or virtual and augmented reality), but with two significantly different premises: first, it is much broader and more visible, and second, there is an expectation that, even after the hype subsides, its impact on society will be lasting. While blockchain implementations like Bitcoin are now widely known but have fallen short of expectations and hopes, AI has, in just two years of hype, managed to fundamentally change how technology is perceived and used across media, government, business, and among private individuals.

The topic of AI is by no means new. One of the earliest research papers on the subject dates back to 1950. Titled "Computing Machinery and Intelligence," it was written by Alan Turing (Turing, 1950). Since then, research in this field has become increasingly active, and for many years, AI has been used in a variety of applications: from detecting terrorist financing in banking transactions, to predicting industrial equipment failures, to forecasting business developments using business intelligence systems. AI has long been an integral part of many digital tools.

The major breakthrough that led to the current state of AI systems was a scientific paper with the unassuming title "Attention Is All You Need," published in 2017 by eight Google researchers (Vaswani, 2017). Building on a 2014 publication, it described a novel deep learning architecture. The architecture, known as "Transformers," is the core of

large language models (LLMs) such as ChatGPT and others. These large language models can understand and generate natural language (how this works is explained in Sect. 1.2). At the end of 2022, the American company OpenAI introduced ChatGPT, a chat-based AI that is essentially an LLM. From that moment, the topic and its possibilities entered the public consciousness.

Given the prominence and apparent advantages of AI, it is important for companies to define their own approach to it. Completely ignoring the topic is a fatal mistake. AI already has a significant impact on businesses. While it will continue to evolve and change, it is not going away. Ignoring it would mean falling behind, missing out on opportunities, and leaving competitive advantages to others.

However, there are two different approaches to engaging with AI. Companies can invest heavily in innovation and experiment with new platforms and technologies. This "play to win" mentality can yield many competitive advantages, but it also requires a certain tolerance for failure, as the rapid pace of development means that AI tools can appear and disappear almost daily. Companies must be willing to take risks, make mistakes, and learn from them, as they are operating in largely uncharted territory. Alternatively, companies can focus on adopting proven solutions, leaving the mistakes and pitfalls to others. With this "play to not lose" approach, organizations follow the trend at a more moderate, but suitable, pace and take on less risk.

Whichever path a company chooses, the entire topic of AI should not be ignored, as it will have a profound and lasting impact on society, government, and business worldwide in the coming years.

1.1.1 The Breakthrough of AI for the Mass Market

If AI itself is not a fundamentally new field, why has it recently triggered such a hype? Unlike previous AI systems, language models are generative—that is, they create something new. In the case of language models, this means text, but the field of generative AI is much broader. Today, music, images, and photos of high quality can be generated simply by describing in natural language the desired outcome.

Traditional AI systems learned from data, mostly numerical in nature, and drew conclusions from it. Implementing such systems can be quite technical and difficult for the interested layperson to grasp. There have been many business use cases, but AI often operated in the background—it was neither particularly noteworthy nor was there much interest in how results were actually determined in specific applications.

With the advent of modern generative AI, perceptions of AI have changed dramatically. Systems like Claude by Anthropic for chat and text, Midjourney for image generation, or Veo by Google for video creation remain highly complex technically, but thanks to integrated language models, they can be operated using natural language. Instead of writing code and training data models to eventually produce human-like text, users today can simply describe a task or goal in their own words and receive an appropriate response.

Many knowledge-based tasks can now be easily supported—or soon even fully automated—by LLMs. For companies, this type of AI offers a significant return on investment: depending on the system, costs are manageable and gains in efficiency and quality are immediately measurable. Moreover, due to their ease of use, such systems can be widely deployed across organizations.

As a result, companies can act more quickly in their business processes and ultimately reduce the costs of value creation. This relieves existing staff and supports them in their routine tasks.

1.1.2 The Limits of Artificial Intelligence

AI is not a panacea, and its many advantages are offset by numerous disadvantages. Fundamentally, AI is a complex technology that, when operated in-house (e.g., in a company's data center), may require significant expertise and infrastructure depending on the scenario—even though advances in AI systems, tools, and platforms have made implementation considerably easier.

A far more significant aspect is the required resource investment. The technology primarily relies on performing large volumes of

mathematical operations (e.g., matrix multiplications) at high speed. Graphics processing units (GPUs) are particularly well-suited for these types of operations, as rendering computer graphics is also largely mathematical, and GPUs are specifically optimized for this purpose. As a result, since the release of ChatGPT, global demand for GPUs has increased significantly. Since GPU production requires rare earth elements, which are sometimes mined under challenging conditions, the purchase and operation of proprietary AI using GPUs is a critical issue for companies that voluntarily or mandatorily fall under the Supply Chain Due Diligence Act. In addition to hardware, substantial energy is also required—so much so that operators of AI systems, such as Microsoft, have at times leased nuclear power plants to supply their data centers with exclusive electricity. In the context of many companies' sustainability initiatives, the use of AI can certainly be factored into the calculation of the CO_2 footprint.

Beyond these external factors, the technology itself also imposes various limitations that should be recognized. For generative AI such as OpenAI's ChatGPT, Anthropic's Claude, or Google's Gemini, a significant factor in the models' capabilities is the sheer volume of training they have undergone. This training does not simply involve feeding information and data into the model and having it be trained. First, training data must be accompanied by metadata so the model can understand what the information means. Second, the results generated in response to queries must be evaluated for quality (accuracy, clarity, coherence, and more). The metadata in the training data is produced through a process known as labeling; input data is described with labels (short texts). For example, segments of images might be labeled "dog," "statue," and "white," while texts might be classified with labels such as "poem," "poetic," and "dramatic." The AI learns to understand content through these labels.

The "problem" here is that this labeling cannot be performed by machines. Companies that train AI systems therefore employ people—often in developing countries—to label the training data. While this may seem unremarkable for examples like photographs of animal statues or poems, it is important to understand that these successful, large models have been trained on all freely available knowledge on the

internet. Unfortunately, this includes not only positive content but also a great deal of negative material, making the labeling of training data a demanding and sometimes psychologically taxing task. The use of generative artificial intelligence thus also has an ethical dimension.

Another aspect of training is "bias." Depending on the selection of training data, the model may inherently possess biases. Just as humans do, such a model knows the world only as it has seen it so far. For example, if the model has only been trained on information about animals, there may be no humans in the model's world. This is a very obvious example. It becomes more problematic when such bias is hidden in the training data and goes unnoticed in small ways, only to become apparent during actual use. As a result, certain groups of people may be unintentionally discriminated against, or calculations may lead to incorrect conclusions.

It is also important to be aware of the limitations when using generative AI. Ultimately, new content (texts, images, etc.) is generated through the sophisticated use of probabilities (see Sect. 1.2). What the model generates may appear correct but can be factually incorrect. It is therefore recommended to use AI systems only for purposes where one can personally verify the plausibility of the results. This also means that models like ChatGPT are not search engines and can generate answers that look correct but are factually wrong.

This type of error is also referred to as "**hallucinations**." In such cases, the LLM invents information as part of its response and incorporates it without any special indication. Numbers, data, facts, names, or even entire contexts may be fabricated. This occurs because AI systems are designed to always provide an answer—even if they do not know the correct one in a given context. The fabricated information is not specially marked or highlighted. This makes it necessary to check any AI-generated responses for factual accuracy before further use. Hallucinations are a general problem of AI systems and occur in all models and from all providers.

Another issue users must be aware of is the so-called **bias**. This refers to content shifts in responses in a (potentially) incorrect direction, resulting in a distorted representation. For example, prejudices or false statements may appear in AI-generated responses without being flagged

as such. This class of error is rooted in biases and imbalances in the training data. For instance, if information about successful female applicants is underrepresented compared to men in a given training dataset for a recruitment process, this will be reflected in biased responses to later queries, potentially resulting in preferential treatment of male applicants. Avoiding this is extremely complex, as it requires ensuring even representation of all data that may be used for future responses before the AI system is even trained.

Currently, extensive research is being conducted to eventually eliminate these errors entirely (if that is even possible) or at least to mitigate their effects.

Although the proper use of current generative AI offers many advantages for companies, these systems are still limited and do not yet operate at the level of human capability. The human brain is vastly more powerful, and we work with both conscious and unconscious knowledge, enabling us to create entirely new things—something generative AI (as yet) cannot do.

1.1.3 Use Cases for AI in Companies

The hype surrounding AI is not without reason, as the possibilities and benefits for companies are numerous and impressive. As previously mentioned, "classical," non-generative AI has already been in use for some time. A good example is predictive maintenance: by analyzing production and usage data, systems can accurately predict when maintenance on machines will be necessary. By detecting anomalies (for example, in rotations, temperature changes, or vibrations), faults and excessive wear can be identified and addressed at an early stage. This simplifies maintenance planning and results in fewer operational downtimes for customers.

Similarly, intelligent error detection and correction can already be implemented during production. For instance, image recognition can detect inclusions or cracks during manufacturing and remove the affected part from the production line.

In use cases involving visual classification, the limitations of current AI systems become apparent once again. For example, in waste sorting or the inspection of glass objects, human personnel is still primarily employed, as AI-based processing is too error-prone or costly.

There are also numerous opportunities to leverage the potential of AI in the generative domain. In software development, models are used that can understand programming code and identify and flag errors during the development process. Over time, these systems learn additional errors and approaches specific to the project, continuously improving their error detection capabilities. Furthermore, these models can often directly generate the necessary code to resolve a problem. This can significantly reduce the time-consuming manual process of reviewing and correcting programming errors.

Even before quality control, during the actual programming phase, language models such as Claude or ChatGPT can provide support. With their programming capabilities, they can generate code for various tasks based on a developer's description of the desired outcome. Drawing on the context of existing code, they can also make intelligent suggestions. However, a major breakthrough in software development is still pending, as a comprehensive understanding of an application is crucial, and the context windows of current models are still too limited (for context windows, see Sect. 1.3).

Another typical use case is found in marketing. Since these models can generate text and images of often quite high quality, they are well suited for marketing applications. There are now also tools that use AI to generate entire websites. As marketing thrives on creativity, these models are primarily used as support, taking over part of the workload. In addition to content creation, they can also develop strategies, conduct market analyses, or create topic comparisons.

Given their proficiency in understanding and generating text, it is not surprising that these models offer various applications in document processing. They are well suited for creating product descriptions, summarizing documentation, or producing multilingual content. If

companies wish to offer their manuals in plain language, they can now have them "translated" by an appropriate language model.

These possibilities are primarily inward-facing scenarios, where AI is used within the company and its application may not be visible to customers or suppliers. On the other hand, there are also outward-facing use cases, where, for example, customers interact with AI systems.

The most prominent example is certainly chatbot technology. Whether through a chatbot integrated into the website or via messaging services such as WhatsApp, Telegram, or WeChat, with the right platform, companies can offer intelligent, interactive, and always-available customer support.

A second well-known example is recommendation algorithms, which are used on e-commerce platforms such as Alibaba or Amazon, or for music and video streaming on Spotify or Netflix. By analyzing numerous data points—from wish lists and browsing history to mouse and eye movements—suggestions can be developed and new content recommended.

1.1.4 Effectively Implementing AI in the Company

The brief overview in the previous section already demonstrates that the potential for deploying generative and non-generative AI in companies is extensive. However, it is important not to use AI for its own sake, but to identify meaningful use cases.

For a targeted and successful implementation, the three-step "Crawl, Walk, Run" approach has proven particularly effective. In the first phase (Crawl), the foundations are laid. Initial project experience is gained, knowledge is accumulated, and extensive experimentation takes place. It is especially useful to identify individual use cases ("lighthouses") where the application of AI can be tested. In the second phase (Walk), the experience gained is disseminated throughout the company. At the same time, experimentation continues, for example by identifying and testing new areas of application. The lighthouses from the first phase may already transition into regular operations. In the third and final phase (Run), the resulting implementation project is professionalized

and scaled. The company then finds itself in the midst of an AI transformation.

The entire process should be accompanied by professional change management. It is essential to identify relevant stakeholders early on and involve them appropriately in the transformation process. With topics such as AI, which ultimately concern efficiency gains and employee relief, concerns and fears must be taken seriously, and the changes must be explained and made tangible for each individual.

1.1.5 The Impact on Roles and Skills in Companies

A transformation such as the introduction of AI not only creates new opportunities for value creation, but also requires companies to build the competencies needed to manage its use in day-to-day operations.

At the strategic level, it may be advisable to appoint a Chief Artificial Intelligence Officer (CAIO). This role, situated at the executive level, serves as a cross-functional or staff position that consolidates responsibility for all relevant AI-related topics within the company. The respective responsibilities and positions are then organized under this role.

This includes, for example, the area of legal and compliance. Legislators have defined various requirements for companies that develop, operate, or use AI. Under the European AI Regulation, for instance, employees must be appropriately trained. AI systems developed in-house must also undergo risk classification and, if necessary, comply with strict documentation requirements. As in other areas, companies must respond to regulatory developments and ensure compliance.

Data protection is another relevant area. As in all other parts of the company, AI is subject to data protection regulations. However, there is a caveat: learning models may use the information provided to them for training purposes. As a result, personal data can often no longer be removed from AI systems. This is an important aspect that carries significant risk and must therefore be addressed accordingly.[1]

[1] Please note that this does not constitute legal advice. For questions on legal matters, please consult a legal specialist.

Another important aspect is "operations," or system management. After all, AI systems are ultimately just another form of IT infrastructure, which may be operated in-house or procured externally. They should therefore be subject to the same standards as other solutions and, for example, be centrally managed. Integration with other platforms and tools via appropriate interfaces (such as with AI agents) must also be centrally coordinated and monitored.

In addition to these roles and responsibilities, which typically already exist within companies, at least one additional new position arises alongside the CAIO. Generative AI systems are operated using natural language, which is inherently complex and often ambiguous. Therefore, prompts submitted to such systems must be crafted with high quality. There are basic rules, techniques, and proven structures for producing optimal results. This task is handled by "prompt engineering," which involves developing and testing prompts, as well as training employees in how to formulate effective queries. This topic will be discussed in detail later. First, however, further technical details on the structure and functioning of LLMs will be provided.

1.2 Large Language Models and Generative Pre-trained Transformers

The apparent magic of language models such as ChatGPT, Gemini, or Claude lies in their ability to understand user input and generate appropriate responses based on it. Ultimately, this is underpinned by "simple" mathematics; based on the query, statistics are used to determine the most likely response for the given context. But it cannot be quite that simple. How is it that two identical prompts can produce different results? How can the system generate plausible answers to even the most complex questions? And how can it provide factually correct statements about the same issue, even when the queries are phrased in completely different ways? The AI seems to understand the user's intent beyond mere word choice.

Behind this lies a model specifically developed and trained to understand and produce human language. Such a **Large Language Model**

(LLM) understands context, patterns, and relationships within texts, enabling it to interpret the meaning of inputs and generate coherent and meaningful responses.

If, for example, you present an LLM with a phrase, it can complete it in a meaningful way. It does so by producing the word that is most likely to follow the existing phrase:

Example
Input and Completions

- Input:
  - The main advantage of AI is
- Completions:
  - The main advantage of AI is the
  - The main advantage of AI is the rapid
  - The main advantage of AI is the rapid processing
  - The main advantage of AI is the rapid processing of large
  - The main advantage of AI is the rapid processing of large volumes of data

If we assume that the LLM always selects the next word with the highest probability, the same input would always generate the same completion—disregarding any (possibly ongoing) training of the LLM and the associated changes in probability values. As a result, the texts would appear very artificial and would differ significantly from those written by humans. The reason is that humans use variability in their phrasing when writing, so there is always an element of randomness. To achieve this sense of creativity and unpredictability with an LLM, there is a parameter called "temperature" (usually a value between 0 and 1) that influences which of the possible next words is selected. Depending on the value, more probable words are used more or less frequently. A low temperature results in simpler and more conservative answers, while a high value (unpredictable and sometimes undesired or exaggerated) produces creative responses.

This raises the question of how the LLM fundamentally knows the probabilities of different phrasings. Why, in the example above, was the

word "is" added first and not the word "cat"? As a first step, one could look at the probability of letter sequences. For each letter in a given language, one can determine its frequency of occurrence and the frequency with which specific other letters follow it. For example, the letters "e," "a," and "o" occur quite frequently, while "ß" is used comparatively rarely. Furthermore, the probability that an "e" follows a "ß" is higher than the probability that another "ß" or a "q" follows. For a given letter, it would thus be possible to generate continuations that are at least mathematically plausible. If you combine this, for example, with average word lengths (by inserting spaces), you could create strings of letters that at least visually resemble text, but in fact are not. The "magic" of LLMs, therefore, cannot be explained at the level of individual letters alone. Essentially, this concept is simply applied to words, subwords, and phrases—probabilities are established between these units.

The underlying concept is a neural network, which can be imagined as a set of interconnected nodes with probabilities assigned to the connections. For example, in the above example, there would be a connection from the word "is" to the word "the," with a high probability value ("weight") assigned to this connection. From the word "the," there are then multiple connections to other words. The number of nodes and layers in which such a neural network is typically arranged is just one parameter that influences the effectiveness and size of the LLM.

By feeding the system with existing texts, user queries, and similar data, these probabilities are continuously trained. However, a significant challenge must be overcome, as there are far more possible texts that can be generated than texts that already exist. The system must therefore interpolate and create "new" content as efficiently as possible. An important technique for this is the **Transformer**, which adds the concept of "attention" to an LLM; the assumption is that not all parts of a sentence are equally important, so the system focuses more on the important parts. This, for example, allows the number of connections in the neural network to be reduced. It also enables the generation of more coherent texts overall. Transformers are responsible for determining this attention and integrating it into the model accordingly.

The components described above are the defining features of systems like ChatGPT. The abbreviation "GPT" in the name stands for

Generative Pre-Trained Transformer. "Generative" because the system is trained and used to generate text. Training is a significant part of the concept; the probabilities described are learned in various ways and over time, but at least partially before use ("Pre-Trained"). Finally, it uses the concept of transformers ("Transformer"), which determine attention and thereby make the system more efficient.

1.3 Tokens and Context Windows

Up to now, LLMs have mainly been described in terms of words—but this is not entirely accurate. Language models use "**tokens**" as their unit; they read text in the form of tokens and, based on the probabilities in the neural network, generate the most appropriate next token. These tokens can be entire words or parts of words (such as syllables)—which means it is fundamentally possible for non-existent words to be generated from parts of existing words.

The concept of tokens is quite important when working with language models. For each chat, a model can only process a limited number of tokens at a time, both for input and output. It is essentially the memory of the AI model in the chat interaction. Like looking through a window, the language model uses this so-called **context window** to view the data provided by the user and the data it has generated itself. When new data is added, the context window "shifts" downward—and everything outside this window is no longer accessible to the LLM. If a prompt, for example, generates a large amount of text, it can happen that previous queries or results are "forgotten." The model can then no longer refer to them, and they are no longer implicitly included in the LLM's processing.

The size of the context window is currently considered the most important limiting factor in scaling language models. Many context windows of commonly available public models currently have a size of about 100,000 tokens (equivalent to approximately 100 pages of text), and some models with ten times that size are already being developed. However, if such models are to become part of everyday life and provide meaningful and comprehensive support to individuals, significantly

more capacity would be required. Calculations have shown that a context window of about 100,000,000 tokens would be necessary to fully capture half a year in the life of a single person. This amount of context not only presents significant challenges in terms of storage (both disk and, especially, RAM), but also poses algorithmic hurdles, as this data must be repeatedly and rapidly evaluated to be incorporated into interactions with the LLM.

1.4 Differences Among Large Language Models

There are now many language models available in various forms, and the development of new and more capable models is constantly advancing. The optimal use case for a given model, as well as the required capabilities, are determined by a range of parameters used in its development and training. The most well-known models powering systems such as OpenAI's ChatGPT, Anthropic's Claude, or Google's Gemini often differ only in specific details.

1.4.1 Licensing and Availability

The code and training data of language models are each subject to specific licenses and may not always be freely available. Various LLM providers release their models and make them fully accessible to the public (e.g., Meta's Llama), while others publish only selected models or none at all. For example, OpenAI offers several smaller, specialized models as open source, whereas the large language models behind ChatGPT are not released.

Open-source models offer two main advantages. First, they can be further developed by the public, allowing both companies and individuals to contribute their expertise to ongoing development. Second, companies often benefit from the ability to operate these models independently with greater ease.

1.4.2 Core Capabilities and Modality

Another criterion is the model's core capabilities. For example, one may consider how well it can understand and analyze text. Conversely, the ability to write creatively or generate text in general is also a distinguishing feature. Additionally, some models are stronger in logical reasoning and problem-solving than others.

In user interactions, models differ in the quality of conversation and dialogue, such as linguistic tone, friendliness, and the typical length of responses.

Language support is also a key aspect. Many models are initially available only in English, while the larger and more prominent ones support multiple languages—including, in some cases, exotic or even fictional languages such as Klingon (known from the Star Trek series).

However, language models are sometimes capable of understanding and generating more than just text. Modern, large LLMs are **multimodal**. They can accept and process text queries, analyze various types of documents, and incorporate this information into their responses. Some models can also interpret images, recognize text and excerpts, and utilize this information accordingly. Interaction is no longer limited to text, as certain language models can understand spoken language and respond audibly—in different languages, tones, and dialects. The extent to which these capabilities are developed also differentiates the models.

1.4.3 Knowledge Base and Currency

The strength of modern language models lies in their extensive knowledge base, which they draw upon during interactions—and this knowledge base varies significantly between models. The quantity and quality of training data have a significant impact on an LLM's performance. The more a model is trained on data from a specific contextual domain (e.g., geographic or socioeconomic), the more its responses will reflect that context. The broader, more comprehensive, and more "objective" the training material, the more generally applicable the model becomes.

It is also relevant to consider when the so-called "**knowledge cutoff**" occurred. This refers to the point in time up to which the model was trained on available data, including, for example, news sites. If a model was trained with information up to a certain date, any information generated after that date (such as news or events) is unknown to the model. In this context, it is also important to distinguish what measures have been taken in models to address bias and hallucinations.

1.4.4 Interaction Capabilities

In practice, LLMs differ in how much of their own and provided context they can understand. Personalization options are also relevant here—can users specify settings (language, tone, etc.) or provide personal information for the model to use?

Furthermore, language models can be differentiated by their ability to understand instructions and deliver consistent, coherent responses based on them. Adaptability to user needs is therefore an important criterion.

1.4.5 Practical Limitations

When comparing language models, it is also important to be aware of the limitations of each system. A key factor is the previously mentioned context window, which greatly influences interaction possibilities. For example, a model that excels at analyzing documents is nonetheless severely limited if its context window is too small to process large documents.

It is also important to compare how quickly queries are processed and answered. This depends partly on the infrastructure on which the model is run, but also on the model itself. Current language models that run on a user's own computer may offer extensive capabilities, but are often so slow in execution that they are impractical for everyday use.

Another consideration is the thematic boundaries of the model and how it handles sensitive topics. Many models have restrictions in place to prevent the generation of certain types of information. For example,

most models will not provide answers on how to manufacture poisons or offer medical advice for issues such as depression.

1.4.6 Use Cases

Finally, LLMs are differentiated by their designated applications. This includes, for example, the ability to understand and generate programming code. Models used in software testing may not be able to generate (good) code, but are effective at identifying programming errors in existing code.

Language models trained solely on a company's internal knowledge are better suited to supporting employees with internal queries than to being used in customer support for handling external customer inquiries.

Another example is the ability to integrate with other AI systems and software tools. These capabilities form the basis for creating autonomous AI agents (see Chap. 6).

Another aspect is the model's ability to adapt to different application scenarios. While some models are strictly tailored to a single use case and perform well only in that context, others can be flexibly adapted to deliver high-quality responses for specific requirements.

1.4.7 System Prompts

The response behavior of (generative) AI systems, and especially LLMs, is controlled by what is known as a **system prompt** (Ramlochan, 2024). The system prompt is a very long and detailed description of the desired behavior of a given AI system. It is written by the respective company during system development and is automatically prepended and executed with every subsequent user request (the so-called user prompts). The system prompt is always hidden from view. Often, the system prompt of a given system is not publicly accessible and is sometimes even treated as a trade secret. Other companies publish and comment on their system prompts to provide greater transparency for users.

The system prompt provides the AI system with various guidelines for responding to user queries. It can also be used to define boundaries that the system must not cross. For example, ethical rules and legal requirements can be specified. In this way, the behavior of the system is directly defined and controlled, without the user being able to influence it later on.

The system prompt can also be used to specify the creativity of responses as well as the system's general role and tone. This gives a given system its recognizable "character" in subsequent interactions.

System prompts are often concluded with various checks and plausibility tests. In addition, checks for correct spelling and grammar are usually included at this stage.

Overall, system prompts control the behavior, language, and output of AI systems.

In the remainder of this textbook, only user prompts will be considered. These are the queries that you—as the user—submit to the AI system.

1.5 At a Glance

- Artificial intelligence has been known as a technology for many years and is applied in a wide range of systems.
- AI-based systems, and especially generative systems that produce text, images, and other media, have made significant technological advances in recent years and now offer substantial optimization potential for businesses.
- Despite all the advantages, these systems also entail risks and limitations that companies need to be aware of. Data protection is a particularly important consideration in this context.
- When introducing AI in a company, the "Crawl, Walk, Run" approach is recommended. This involves implementing the technology gradually and with care.
- To ensure optimal use within the company, employees must be trained in the correct application of AI technology. In addition, staff

should also be educated about the limitations and risks to avoid potential data protection violations.
- The EU AI Act regulates the rights and obligations of both providers and users of AI systems. For example, it mandates compulsory training for personnel.
- Large Language Models (LLMs) such as ChatGPT, Claude, and Gemini are based on transformer technology and can generate highly detailed and in-depth texts in response to specific prompts. While they differ in certain functionalities, at their core they all offer very similar capabilities.
- The fundamental functionalities, characteristics, and limitations of an LLM are defined in the so-called system prompt. This system prompt is typically not visible and is sometimes treated as a trade secret. The system always inserts this prompt before the user's prompt and executes it every time.

2

Fundamentals of Prompt Engineering

Abstract This chapter introduces the fundamentals of prompt engineering. To this end, the Valisory Prompt Quality Model (PQM) is first presented as a means of assessing the quality of prompts. Subsequently, the basic techniques for effective prompting are introduced step by step, and their implementation is illustrated using practical examples.

The choice of model in the LLM already has a significant impact on the quality of input comprehension and the results produced. However, regardless of the model's capabilities, crafting good and precise prompts—that is, the quality of the query to an AI system—is essential for effective use. This helps prevent misunderstandings and makes the intention clearer. In addition, the results can be structurally tailored much more closely to one's own expectations. The following sections explain various fundamentals for effectively formulating prompts for a language model.

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_2

2.1 The Concept of Prompt Engineering

Prompt engineering is a specialized field within AI that focuses on optimizing input prompts for language models such as ChatGPT, Claude, or Gemini. It encompasses various techniques for the targeted, precise, and effective formulation of prompts.

On the one hand, the need for prompt engineering can be illustrated by analogy to interpersonal communication. The human brain is vastly more powerful than current language models. Nevertheless, training in communication is a natural part of many educational and professional development programs. As humans, we can quickly grasp the intended meaning in standard situations, even if the content is awkwardly phrased, unclearly spoken, or expressed out of context. Still, we invest in improving our communication to reduce effort and prevent misunderstandings.

On the other hand, language models are generally already quite capable of correctly interpreting and effectively responding to queries that are not clearly formulated, lack context, or do not specify the desired outcome. If the results do not meet expectations, they can be interactively adjusted or improved with the model; however, eliminating these optimization cycles would make the work significantly more efficient.

This optimization is the fundamental goal of prompt engineering. By structuring prompts or using specific formulations, the outputs of the LLM are guided, thereby increasing quality and reducing the need for post-processing. In essence, the capabilities of the language model are fully leveraged to achieve the best possible results.

The concept of prompt engineering is based on three interrelated levels. The foundation consists of requirements for linguistic clarity, correct definition of expected outcomes, and working with referenced texts or documents. Building on this, various frameworks can be used to organize these fundamentals into a specific structure dictated by the respective framework. The third level comprises various techniques related to working with examples or improving multi-step results, which can be combined with both the fundamentals and frameworks. All three levels are explained in detail with concrete examples in the following chapters.

In addition to simply writing prompts, prompt engineering also involves quality assurance and testing of queries. While for simple and interactive use cases it is usually sufficient to employ various prompt techniques to produce good results, there are scenarios such as AI agents (see Chap. 6), where prompts cannot simply be changed and results cannot be iteratively refined. In such cases, different prompts are written for a given task and compared in terms of their effectiveness.

A level above this is the aspect of multi-LLMs and prompt security. When multiple LLMs are combined, effective prompts must be crafted for each system, and the integration of the LLMs must be carefully planned. In terms of security, prompt engineering addresses the question of how, for example, chatbots can be prompted to prevent misuse. This area also deals with overcoming the artificial boundaries set for language models, for instance, to extract information on prohibited or restricted topics.

2.2 Assessing the Quality of Prompts

The use of prompt engineering techniques to optimize queries to language models for improved results necessitates the ability to assess the quality of both the prompt and the outputs. Without a benchmark, even the best-sounding techniques may not deliver the desired benefits and may only create additional effort.

Measuring quality is no easy task. Starting from the situation where a language model's output does not meet expectations, the question arises whether more was demanded of the model than it is technically capable of delivering. If the limits of what is technically feasible have been reached, no prompting technique, however sophisticated, can improve the results. For example, a text generation model fundamentally cannot perform calculations and will produce incorrect results or draw false conclusions from numerical data. Regardless of how a query is phrased or which prompt framework is used, it will not lead to correct results.

If you ask an LLM to perform calculations, you may (depending on the model) receive correct answers. Simple arithmetic tasks within a small numerical range (for example, up to 100) can often be answered correctly based on data from the training material. For more complex tasks, ChatGPT, for instance, typically generates and executes a Python program to solve the problem—in such cases, the results are also correct. Beyond that (and especially if you require that no Python be used), it becomes clear that the language model is not a computational model.

Setting aside the technical limitations of the model, quality can be determined simply by comparing the output to the set expectations. If these are not met or only partially fulfilled, the prompt should be optimized until the necessary level of expectation is achieved. To avoid arbitrary changes, improvements should be made in a structured manner, meaning the quality of the prompt should be assessed based on clearly defined criteria. However, it should be noted that even the best set of quality parameters is to some extent arbitrary, as the exact path from query to output is so complex that there is no linear relationship between quality criteria and results. Nevertheless, such an approach helps to use LLMs more effectively.

The Valisory Prompt Quality Model (PQM)

The Valisory Prompt Quality Model (PQM) is a multidimensional quality framework designed to assess the quality of prompts submitted to generative AI models. It is applicable not only to language models but also to all other types of generative AI, such as music, video, or images. To determine the quality of a prompt, it is evaluated according to four criteria: relevance, creativity, language, and target audience, with each criterion further divided into three distinct aspects (see Table 2.1).

The **relevance** aspect assesses the extent to which the prompt's wording is fundamentally appropriate to the task objective. This involves evaluating the question formulation, specificity, and ultimately, comprehensibility. The *question formulation* aspect considers how clearly the prompt is articulated and to what degree it directly requests the desired information, results, or actions. This includes both linguistic accuracy and clarity of instructions regarding the expected outcomes. The

Table 2.1 The Valisory Prompt Quality Model

The Valisory Prompt Quality Model				
Criterion	Relevance	Creativity and Originality	Linguistic Quality and Style	Target Audience Appropriateness
Factors	1. Question Formulation 2. Specificity 3. Comprehensibility	1. Innovation Potential 2. Inspiration 3. Variability	1. Clarity 2. Adaptation to Style 3. Structure	1. Understanding of Target Audience 2. Addressing the Audience 3. Usefulness

specificity of the prompt refers to how concretely the requirements and stated objectives are defined to generate relevant responses. Finally, the *comprehensibility* aspect examines whether the prompt is written in a way that allows the language model to interpret the request correctly and remains easily understandable, even for the author.

The **creativity and originality** category evaluates the degree to which the variability of results is constrained. First, it assesses the extent to which the model is encouraged to develop creative solutions (*innovation potential*). Next, the *inspiration* aspect measures how much the prompt stimulates the generation and discovery of original ideas and highlights perspectives that are not trivial or obvious. The third factor, *variability*, evaluates the scope the prompt provides for different answers or solution paths.

The next dimension, **linguistic quality and style**, addresses the effectiveness of the prompt's formulation. The *clarity* criterion examines how unambiguous and precise the prompt is linguistically, aiming to avoid ambiguous or misleading wording. Additionally, it assesses whether the phrasing is appropriate for the intended purpose (*adaptation to style*). For example, if a prompt is written in a technical, formal, or informal manner, the language model will adapt accordingly. The third aspect, *structure*, considers whether the prompt is logically organized and facilitates the development of a structured response.

The first three criteria primarily focus on the communication between the user and the language model. The final criterion, **target audience appropriateness**, additionally considers how well the

prompt's results align with the needs of the intended audience. This begins with evaluating the extent to which the prompt incorporates and clearly explains the target audience's prior knowledge, interests, and needs to the LLM (*understanding of target audience*). The criterion also includes how well the prompt's tone and address are tailored to the audience (*addressing the audience*). While the linguistic quality and style criterion focuses on the prompt's formulation itself, this dimension emphasizes the use of explicit stylistic instructions for the target audience. The final aspect examines how useful the type of response elicited by the prompt is for the target audience, how practical it is, and whether it is generally of interest (*usefulness*).

With these twelve factors of the Valisory PQM, prompts for AI models can be comparatively analyzed in terms of their quality. Although the evaluation of individual aspects is inherently subjective, the model provides a robust framework for quality assessment and highlights areas where there may be potential for optimization.

The Valisory Prompt Quality Model (PQM) for Evaluating Prompts

Relevance

1. **Question Formulation**: Is the prompt clearly formulated and does it specifically request the desired information or action?
2. **Specificity**: Are the requirements and objectives of the prompt specific enough to generate relevant responses?
3. **Clarity of Understanding**: Is the prompt simple and clear enough for the model to correctly interpret the request?

Creativity and Originality

1. **Innovation Potential**: Does the prompt encourage the model to go beyond general knowledge and find creative or innovative solutions?
2. **Inspiration:** Does the prompt inspire original ideas or perspectives that are not trivial or obvious?
3. **Variability**: Does the prompt allow sufficient scope for different answers or solution paths?

Linguistic Quality and Style

1. **Clarity**: Is the prompt clear and precise, without ambiguous or misleading wording?

2. **Style Appropriateness**: Is the style of the prompt (formal, informal, technical, etc.) appropriate for the objective and the audience?
3. **Structure**: Is the prompt logically structured and does it encourage a structured response?

Target Audience Appropriateness

1. **Audience Understanding**: Does the prompt take into account the prior knowledge, interests, and needs of the target audience?
2. **Addressing the Audience**: Is the tone and directness of the prompt appropriate for the target audience?
3. **Usefulness**: Is the type of response sought by the prompt practical and relevant for the target audience?

2.3 Linguistic Formulation of Prompts

When formulating prompts, it is important to be as precise as possible. While comprehension capabilities are extensive, our language is by no means always unambiguous. Therefore, try to clarify any potential ambiguities or double meanings. This also applies to technical terms, especially if they are uncommon or may have different meanings depending on the context. Consider, for example, the term "account," which means something different in accounting than it does in system administration or finance. Also keep in mind that the LLM does not have access to your informational context. It may be useful or even necessary to include information in the prompt that seems trivial or obvious to you.

Use colloquial language sparingly and make use of punctuation marks—they can significantly alter the meaning of language. Whenever possible, phrase your prompts positively. Instructions such as "Focus on the strengths" consistently yield better results than "Do not focus on the weaknesses."

And despite these recommendations and practices, you do not need to communicate as if you were speaking to a person; phrases like "please," "thank you," or even "hello" do not provide any real advantage in the interaction.

In the following sections, practical examples will be provided for all the tips on prompting discussed. To get the most value from this

book, we recommend that you try out the examples in an LLM of your choice. This way, you will gradually become accustomed to the techniques presented. It is also advisable to apply the explained techniques to examples from your own life or work. This will allow you to see results quickly. Finally, if you compare simple prompts and their results with those created using good prompt engineering, you will be surprised at how much better, more detailed, and more precise the LLM's answers are.

You will see: practice makes perfect. And since prompt engineering does not require learning a new programming language, but rather a thoughtful use of your natural language, certain linguistic constructs, and processes, you will quickly see results and prompt engineering will soon become second nature to you.

2.4 Defining the Output Format

The output format of the language model can be influenced in a variety of ways. Consider in which structure and formatting you need the result, and communicate this to the model accordingly. For example, you can have results output as a list, have them sorted in a particular way, include specific information, or explicitly exclude certain details—such as if the LLM would otherwise add them by default.

> If the result is not quite right, that's not a problem. You do not need to repeat the entire prompt—simply tell the model which parts of the result should be adjusted.

Let us illustrate this with an example. We ask ChatGPT to create a list of the ten largest cities in the world.

Example

Create a list of the ten largest cities in the world.

The list can now be transformed and adapted to your needs in various ways. For example, with the following prompt, you can have it sorted in descending order, omit the numbering, and add the respective countries for each city.

Example
Prompt Example

Sort the list in descending order based on population. Omit the numbering. Add the respective country and the area in square kilometers for each city.

Since the LLM will make its own assumptions about the presentation and formatting of the answer if not given specific instructions, you should also provide appropriate guidance here. Otherwise, you may receive your result as a table, a numbered list, or as continuous text.

Example

Format the result as a sorted table and write the names of the individual cities in bold. All numbers shown should be rounded up or down to the nearest ten and displayed without commas. The table should be structured as follows:
Table header: The Largest Cities in the World
Columns: Number, City Name, Country, Population, Area in sq km

But the formatting options do not end there. Large language models are capable of converting texts, data, and entire tables into almost any desired format. For example, you can have the table from the example converted into "CSV" format (comma-separated values). This format is supported by all common spreadsheet programs, allowing you to process entire tables directly in Excel, for instance.

Formatting in HTML with accompanying CSS is also possible with just a single prompt. This allows the table to be embedded directly into a website and viewed in any browser.

The formatting possibilities are virtually endless. So before you submit your prompt, take a moment to consider how you want the result

to be structured and formatted. Then describe your requirements as precisely as possible. This way, you make it clear to the LLM exactly what it needs to do, and you save yourself the time of having to further adjust the answer afterwards.

2.5 Referencing Texts

It is common to integrate existing text into your prompt, for example, to summarize it, rephrase it, or use it as an example within the prompt. Simply appending these texts to the prompt can work, but it may lead to various unwanted side effects. Therefore, there are different techniques to distinguish between the actual prompt and the attached texts.

Multiline texts can be delimited with three quotation marks (“““) or hashtags (###) before and after the text. These delimiters should each be placed on their own line.

Example

Rewrite the following haiku in the style of Franz Kafka:

"""

Words show the way,

In silence the soul speaks,

Language bridges the world.

"""

Especially in more extensive prompts, it may be necessary to process multiple texts or represent recurring situations (such as a dialogue). For this purpose, sections can be used in the prompt to reference text passages later. Sections have a name, followed by a colon and the context to which they refer. The context can be a sentence or a multiline text. It is also possible to refer to attached files in this way. It is not the case that the language model recognizes the syntax of sections and processes the following information as a fixed language construct. Rather, the intention is understood here as well. Therefore, the model can

handle both fixed data (such as a direct quote) and content that requires interpretation.

Example
Here are two statements. Rephrase the technical response in a friendlier manner.

Request statement: The system is not available.

Technical response:

###

There is an irregularity in operations.

###

2.6 Specifying Solution Approaches

Effectively formulating and structuring prompts already has a significant impact on the quality of the results. However, if the outcome does not meet your expectations, the LLM implicitly makes assumptions, or requirements are not met, it can help to structure the solution approach. Explain to the model how it should proceed to arrive at a solution. You can also ask it to explain the intermediate steps, allowing you to monitor the quality of the process.

Try this by writing a single prompt for the following scenario: The language model is to develop an Instagram post for a company of your choice. In the first step, possible target groups should be identified, then the best target group should be selected. For this target group, a SWOT analysis (Strengths, Weaknesses, Opportunities, Threats) should be conducted, using only four factors per segment—this limits the effort enough so that the response is not too lengthy while still being meaningful. Afterwards, the LLM should derive the primary *pain point* of the target group. Finally, in the last step, an Instagram post should be developed, with both the image and the caption explained.

The result of such a prompt is quite comprehensive, as the intermediate steps are each explained by the LLM, making both the partial results

and the final outcome easy to follow—the model's reasoning thus becomes transparent.

> Something not right? You can repeat the process or have only parts of the intermediate steps adjusted.

2.7 The Interview Format

The previous prompts were essentially self-contained and provided the LLM with specific instructions as well as all relevant information. While this is a common and intuitive approach, more interaction—especially in detail-rich situations—can significantly improve overall results. To achieve this, the model is given the opportunity to ask the user for information.

In the basic version, the questions to be asked are specifically defined and included in the prompt. For example, the LLM can be instructed to ask for details about a target group. At the appropriate point, it will then ask a corresponding follow-up question, which the user can answer as usual in the chat window. Processing then continues. This can be very effectively combined with the concept of specifying solution approaches from Sect. 2.6.

Including the questions to be asked in the original prompt is based on the idea that you already know which questions will be relevant during the course of working on the problem. You can also let the LLM decide which questions it considers relevant for answering your prompt by instructing it to ask for missing information and pose clarifying questions.

Below is an example prompt that, with suitable follow-up questions, composes a short LinkedIn post.

Example

Create a short LinkedIn post for me. I would like to report on my current project and highlight what has already been successfully implemented for the client. The post should have a slightly sales-oriented approach and encourage readers to contact me for similar projects.

Ask me relevant questions to clarify the post so that you can create an optimal post for my target audience.

You do not need to explicitly state that your answers to the questions should be included in the response—this happens automatically.

You can also incorporate repetitions by instructing the model, for example, to keep asking questions until all relevant information is available.

2.8 At a Glance

- Prompt engineering refers to the formulation of effective queries to generative AI-based systems, promising optimal responses.
- With the help of the Prompt Quality Model (PQM), the quality of AI queries can be systematically evaluated. This provides the basis for optimizing prompts.
- Prompt engineering does not require learning a new programming language. It simply involves the deliberate use of proven natural language phrases, constructs, processes, and instructions.
- Prompts should include a very concrete description of the desired result to ensure that the implicit expectations for the outcome are met.
- An AI system can independently search for possible solutions to a given problem. However, with a suitable prompt, clear instructions can also be provided.

3

Frameworks for Prompts

Abstract This chapter presents proven approaches that serve as structural guidelines for creating high-quality prompts. These so-called prompt frameworks provide specifications that can be used as linguistic and structural references when formulating queries for AI systems. Throughout this chapter, various prompt frameworks that have demonstrated strong results in practice are introduced. The application of each framework is illustrated with practical examples.

With the fundamentals of prompt engineering, a great deal can already be achieved in terms of quality with the respective language model. Building on this, various frameworks have been developed over time that prescribe a specific structure for prompts in order to obtain better responses. These structures are by no means an end in themselves; for many prompt frameworks, there is scientific research that demonstrates the effectiveness of the respective approaches. The following section presents several common frameworks for writing effective prompts.

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_3

3.1 Prompt Chaining

Prompt chaining is a very simple yet highly effective method that can significantly improve the results produced by a language model. In prompt chaining, you mentally break down the task into several steps and then work through these steps one by one with the LLM. At each intermediate step, you review the results, make adjustments if necessary, and then proceed to the next aspect. This approach is essentially similar to the advanced technique of chain-of-thought prompting (see Sect. 4.3). However, prompt chaining typically requires more manual effort and often takes more time to achieve comparable results.

Since prompt chaining may involve extensive work with the language model on a single task, it is important to pay attention to the size of the context window (see Sect. 1.3).

You can also combine prompt chaining with other prompt frameworks.

Let us use the example of writing an article on the topic of bicycles. We want to use the model to identify relevant topics, structure them logically, and then have the article written.

In the first step, the model is asked to identify relevant subject areas. Explanations are deliberately omitted to focus on the essentials. The model is provided with sufficient information about the objective so that it can complete the task.

Example

Your goal is to write an article about the mechanics of a standard bicycle. We will develop the article step by step.

First, give me an uncommented list of relevant topics.

Depending on the LLM, the length and detail of the resulting list will vary. If topics are missing, the model has identified irrelevant aspects, or

is completely off track, you can intervene early by prompting it to make adjustments.

If the initial approach is suitable, you can proceed with drafting the article. Not all identified topics may be covered in the article. Therefore, a selection should be made, and the model should be asked to justify its choices so that the process remains transparent.

Example
Select and justify the five most relevant topics.

If the structure is appropriate, you can now focus on the content of the article. Before the article is actually written, it is advisable to develop a more detailed outline of the topics. For example, the LLM could be prompted as follows:

Example
Create a table of contents. Arrange the topics in a sequence that provides a logical and coherent flow. Add an introduction and a conclusion. For each point, write three bullet points describing the content.

At this stage, content changes can still be made easily before too much time is invested in writing the actual article. For example, the LLM could be asked to use a storytelling framework or to suggest how meaningful chapter transitions could be implemented. Finally, the article can be completed.

Example
Now, based on the points developed, write the article with approximately 300 words per topic.

Prompt Chaining with Expert Levels

A variation of the prompt chaining technique is working with expert levels. Here, the language model is also asked to generate a solution to a problem. Then, the concept of a rating scale is introduced, the initial solution is placed at the first level, and in two subsequent prompts, a slightly better and then the best possible solution are requested.

To illustrate this, the planning of a webinar is used. In the first step, a solution is requested without any particular structure or additions.

Example
Develop a three-step plan for organizing an optimal webinar. Present each step as a short, concise sentence.

The result of this prompt will be simple, yet comprehensive and meaningful. Each step will be briefly outlined in exactly one sentence, without going into too much detail. Next, the idea of a rating scale is introduced and this result is set at the first level. The LLM is then asked to produce a level 2 result. Without defining what distinguishes a level 1 result from a level 2 result, it cannot be expected that the next result will be significantly better or more detailed. The model works here with assumptions about quality, which are not of significant relevance for the implementation.

Example
Assume this solution is rated 1 out of 10 on a scale. Now develop a solution that corresponds to level 2.

The result produced at this step is typically not far removed from the solution in the first step. It may include more aspects and be somewhat more detailed, but otherwise will be quite similar. To achieve a significant leap in quality, the LLM is then asked to develop a solution at the fictional level 10. Again, it is not defined what constitutes a rating of 10. However, the model now has two data points, one for level 1 and one for level 2, from which it can infer what a much better result might look like.

Example
Now develop a level 10 solution.

The result of this final step will differ significantly from the previous two and will contain more information. Since the initial prompt restricted the response to a single sentence, the model will, for example,

use more complex sentence structures and string together ideas with semicolons.

After the first result, you can also specify that the current text is a "beginner-level" answer and ask for a revised response at the "intermediate level." This result will again be more detailed and comprehensive. In a final step, you can then request the "expert level." You will see that the quality of the result will improve dramatically, and the text will gain in length and depth. In practice, it has been shown that further optimizations of this kind do not yield significantly better texts. Overall, this relatively simple approach can significantly enhance the quality of results.

3.2 Role Prompting

To significantly influence the context of the task and its results, the model can be assigned a specific role for processing. As with prompt chaining (see Sect. 3.1), role prompting is a very simple yet extremely effective means of improving the quality of results.

For this, the LLM is assigned a role from whose perspective it is to approach the task. Such a role can, for example, be a job description or a function within a company, such as a software developer or an accountant. Simply imagine that you could pose your query to the best expert for your specific question and assign exactly this role to the LLM. Both real, well-known individuals and fictional characters can be chosen, or abstract role descriptions can be used.

Examples of different roles:

Example

- Answer the following question exactly as the Greek philosopher Plato would have.
- Behave as unemotionally as the Vulcan Spock from the Starship Enterprise and answer the following question with strict logic.
- You are an experienced attorney with over 20 years of professional experience in labor law and will provide me with detailed advice on the following legal question.

If it can be assumed that the model is (sufficiently) familiar with the role, it is not strictly necessary to provide additional information. The role is typically defined at the beginning of the prompt with phrases such as "Act as …" or "You are a …". In practice, however, it makes no real difference where such a formulation is placed within the prompt.

Example
You are an experienced Agile Coach. Develop a one-day workshop for a programming team on the topic of Scaled Agile Framework.

Especially in a corporate context, it can be expected that large language models are already familiar with many common as well as less typical roles. If in doubt, the model will not ask for clarification but will make assumptions, so it is advisable to be precise in defining the role.

If the role needs to be specified more precisely or if the language model is unlikely to have much knowledge about it, it can be helpful or necessary to provide additional information. This might be a more detailed description of the role or supplementary informational material. The role can be described in detail within the prompt text, or a file containing a specific role description can be uploaded.

Example
You are a JAVA software developer. In the attached file "Skills.pdf" you will find an overview of your skills and experience.

Interestingly, specifying the level of experience (for example, in years) can actually lead to better results.

Role prompting is so effective and straightforward that it is a component of almost every other prompting framework. It is strongly recommended to include role prompting as a fundamental building block in all prompts that are more than just a simple question.

3.3 Deductive Problem Solving

The prompt technique of deductive problem solving, similar to prompt chaining, proceeds in two steps. In the first step, the model develops a general plan or scenario. In the second step, this is then applied to the user's specific situation.

This approach essentially mirrors how humans tackle major problems. A basic model is developed in thought and then naturally applied. For example, when planning an event, one might initially ask: "What are the fundamental steps in planning an event?" In this sense, this prompt technique breaks the problem down into two steps.

3.3.1 Developing a Metamodel

The first step in deductive problem solving is to develop a general approach at the meta level. For this, the language model is asked—independent of the user's context—to design a general solution that addresses the problem as optimally as possible. This solution can then be refined step by step until it reaches a state that is appropriate for the actual task at hand.

Example
Develop a strategy for planning an employee meeting for a company.

3.3.2 Applying the Model

Once the metamodel is complete, it is applied to the actual problem. To do this, the prompt refers to the general model and describes the situation and objective. At this point, one of the other prompt frameworks can be effectively used for structuring.

Example

Now apply the strategy just described for optimal planning of an employee meeting specifically to the planning of my company's employee meeting. My company has 250 employees at 10 locations in the DACH region. The event is to last one day and will take place on-site at our headquarters in Berlin.

3.4 The R-T-F Framework (Role-Task-Format)

The R-T-F Framework places the model in a specific role before it processes the prompt's task. The abbreviation stands for the three building blocks of the prompt: the role the LLM is to assume (Role, R), the task it is to perform in this role (Task, T), and the required output format (Format, F).

3.4.1 The Conceptual Role of the Language Model in the Prompt

The first step of the R-T-F Framework is to inform the LLM which role it should take on when answering the question. This is done in the same way as described in Sect. 3.2 "Role Prompting."

Communicating the role is straightforward. This is the first instruction in an R-T-F prompt and does not follow any specific syntax. Here are a few examples:

Example

You are a software architect.

Assume the role of a sales coach.

Act as a social media consultant.

It is not always sufficient to specify the role without further information. For example, the role of a sales coach can vary greatly. Who is being coached? Companies? Freelancers? What is their target audience? Is it about closing coaching, or are we in the context of cold calling?

To specify the role more concretely, you can simply add further parameters or context. Again, the more precisely the context is described, the better the results will be.

Example
You are the head of logistics at a medium-sized company. You manage a team of 10 employees and are planning a team event to improve team cohesion.

3.4.2 The Task of the Prompt

Once the role has been defined and, if necessary, further specified with additional context, the second part of the prompt sets out the specific task. There is no special format for this; the prompt is formulated here as it would be without a specific framework.

It is important that the model acts in the previously specified role at this point. Therefore, technical terms or abbreviations can be used here without issue, as long as they are generally understood within the context of the role.

Example
You are the head of logistics at a medium-sized company. You manage a team of 10 employees and are planning a team event to improve team cohesion.

Your task is to plan a two-day team offsite. Activities should be planned in the areas of cooking, sports, meditation, and teambuilding. Each day should start at 9 a.m. and end at 6 p.m.

3.4.3 The Output Format

The third part of the R-T-F Framework defines the format of the results. This could be, for example, a list or a longer text. The approach from Sect. 2.4 is also used here. Again, the more specific the format is defined, the better the results will be.

In practice, the format of the response often does not fully meet expectations. Simply instruct the model to adjust the results accordingly, for example, with or without descriptions, or in the form of a list.

Below you will find a complete sample prompt using this framework. The goal is to have a schedule for a team event created. Thanks to the R-T-F structure of the prompt, the result is organized both in terms of content and structure, making it easy to process further. In the *Task* section, additional information could be added to tailor the prompt even more closely to the situation.

Example

You are the head of logistics at a medium-sized company. You manage a team of 10 employees and are planning a team event to improve team cohesion.

Your task is to plan a two-day team offsite. Activities should be planned in the areas of cooking, sports, meditation, and teambuilding. Each day should start at 9 a.m. and end at 6 p.m.

Develop a schedule for both days. Specify the start and end times, create a title, and provide a brief description for each activity.

3.5 The T-A-G Framework (Task-Action-Goal)

The T-A-G framework focuses on completing a specific task, which is broken down into individual steps and subordinated to an overall goal. The LLM is given the freedom to decide on the format and structure of the results. The name T-A-G stands for Task, Action, and Goal.

3.5.1 The Task of the Prompt

The first part of the T-A-G framework describes the task to be accomplished. The model is briefly and concisely informed of the expectation

and what it is specifically supposed to do. The task description does not need to be exhaustive or absolutely precise, as further information will follow.

Examples of tasks could include developing a marketing strategy, conducting a potential analysis for a product, or writing a proposal. The task should not be too granular, as it will be broken down into steps in the next part. As an example, we will construct a multi-step potential analysis for a fictional product:

Example
Your task is to develop a potential analysis for an innovative fruit juice. The innovation consists of the artificial addition of vitamins and minerals.

3.5.2 The Actions to Be Taken

Once the main task of the prompt has been defined, the second part lists the steps necessary to achieve it (Actions). These steps usually build on each other, starting more generally and working step by step towards the prompt's task.

The LLM will follow the steps in the order provided. Therefore, make sure to organize them logically. Provide all necessary information for each step. Here, too, you can specify requirements regarding structure and formatting. If the action descriptions become too confusing, use structuring elements such as a numbered list for the actions or sections.

You can also instruct the language model to determine the order of the actions itself.

The actions can be effectively combined with the concept of roles from the R-T-F framework.

Here is the second part of the prompt from Sect. 3.5.1. The actions are stated briefly and concisely, without imposing significant requirements on formats or providing many details. The model will often also draw a conclusion from the results, even if this was not explicitly requested.

Example
Briefly identify possible target groups. Select one target group with justification. Conduct a SWOT analysis for this target group with three factors each.

3.5.3 The Overarching Goal

The final part of the T-A-G framework is the goal, which places the previous information under an overarching purpose. This ensures that the language model's approach and the generated results are explicitly aligned with the intended objective. This section also defines additional parameters and constraints.

In the example, a potential analysis is to be conducted based on the selection of a target group and a SWOT analysis. The reason for this is not specified—should the product be rebranded? Does the product already exist? Is a competitor comparison required? Depending on how these questions are answered, the focus of the potential analysis will shift and the results will differ.

In this example, the goal is to identify the target group that is expected to generate the highest revenue. Although the goal is stated at the end, it naturally influences the entire process, and both the selection of the target group and the subsequent analysis are conducted with this objective in mind.

Example
Your task is to develop a potential analysis for an innovative fruit juice. The innovation consists of the artificial addition of vitamins and minerals.

Briefly identify possible target groups. Select one target group with justification. Conduct a SWOT analysis for this target group with three factors each.

The goal is to identify the customer market with the highest revenue potential.

3.6 The B-A-B Framework (Before-After-Bridge)

The B-A-B Framework (Before, After, Bridge) for prompts focuses on the current and target situations. It is used to develop a plan or strategy for achieving a goal. In the first step, the current situation is described, followed by a description of the target situation. The third part explains the approach to achieving the goal.

3.6.1 Description of the Current Situation

The first part describes your current situation. This description can consist of just one sentence. It is also possible to provide more information to offer a clearer and more effective context. The better you outline the current situation, the better the results will be in the end.

As an example, let's consider developing a market entry strategy. A fictional product, in this case a smart toaster, is to be sold to customers in France in the future. In the current situation, the product already exists and is being sold in another country. Here is one way to express this as the Before part of the prompt:

Example

We are developing a smart toaster that is already being sold in Germany and Austria. So far, we are only present in these countries.

Viewed in isolation, this part seems rather awkward. It merely states facts and does not yet indicate how to proceed with this information.

3.6.2 Presentation of the Target Situation

After explaining the current situation, the next part presents the desired target scenario (*After*). The same rules apply as in the previous section: the more information you provide about the goal, the more useful the results will be. The basic goal in this example is to have entered the French market.

Example
We want our product to be present in France.

However, in this form, the goal is rather vague—for example, it raises the question of whether market entry is considered successful as soon as a single smart toaster is sold. In such cases, it may be useful to specify more precisely what is meant. If measurable objectives are defined, the LLM will take them into account when developing the plan.

Example
We want our product to be present in France and to have sold 10,000 units in the first quarter after market entry.

As with the current scenario, this part of the prompt is not very meaningful or informative on its own. At this point, the model only knows your current position and what you want to achieve, possibly including a timeframe.

3.6.3 Development of the Plan

The final part of the B-A-B Framework (*Bridge*) communicates how you plan to bridge the gap between the current and target situations. Here, you formulate the specific task to be accomplished. For example, you can instruct the model to outline a strategy plan to achieve the set goal. You can also have a detailed action plan developed that explains the necessary steps. Additionally, it is possible to have the LLM generate sample interim results to conceptually simulate the implementation.

The fundamental idea of the B-A-B Framework is to have the language model develop the plan for achieving the goal. You therefore tend to formulate in categories and usually do not specify concrete steps. However, if you already know which steps are at least partially necessary to reach the target situation, you can specify them as well. In that case, be sure to indicate that additional meaningful steps should be added.

Here, you can also specify a role. As with the R-T-F Framework, assigning a role allows the developed approach to be tailored more specifically to your situation.

The example in this chapter refers to the market entry of a product in France. A simple Bridge instruction could look as follows:

Example
Develop a market entry strategy for the product that will enable us to achieve the sales target.

At this point, you can also provide additional information that is important for the outcome. For a market entry strategy, the target group is just as relevant as the brand's positioning and identity. While the latter is something the model should be explicitly informed about, the target group can also be determined by the model itself. You can instruct it to conduct a target group analysis, select a target group based on criteria you provide, and then focus the developed strategy accordingly.

Example
We are developing a smart toaster that is already being sold in Germany and Austria. So far, we are only present in these countries.

We want our product to be present in France and to have sold 10,000 units in the first quarter after market entry.

Develop a market entry strategy for the product that will enable us to achieve the sales target. You will find our marketing and brand

strategy in the attached file. Identify a target group for the market entry and conduct a SWOT analysis for this group. Present the results in a table.

3.7 The C-A-R-E Framework (Context-Actions-Result-Example)

The C-A-R-E Framework (Context, Actions, Result, Example) uses four steps or components to generate one or more outcomes within a clearly defined context. The expected result is specified in the prompt, and one or more examples are provided.

3.7.1 The Context of the Prompt

In the first step (*Context*), the context is communicated in which the subsequent instructions are to be understood. This context can, for example, be the initial situation you are currently facing. Provide as much information here as necessary for the language model to understand your current position. If you want to achieve a business-related outcome, it is advisable to include information about the company, your department, or your role.

> In this framework, you do not define the desired outcome in the context.

Let us illustrate this using a workshop planning example. Suppose you already have a completed workshop that you now want to adapt, for instance, into a leadership seminar. The structure and approach should largely remain the same, but content additions or adjustments may be possible or even necessary.

A possible context could be as follows:

Example
I am a productivity coach for employees and executives in medium-sized companies. I would like to adapt my workshop "Working More Productively – The Basics," which is currently offered to employees, for executives. The target group is C-level individuals who will learn to work more effectively in their daily business.

3.7.2 The Actions to Be Taken

The second part of the prompt (*Actions*) then defines which actions are to be carried out. If the path to the goal is already known, you can specify exactly what should be done, in what order, and to what extent.

If there are still open points in the process, the LLM can also be instructed to independently take reasonable steps. For transparency, you can require that a plan is first developed and explained before the individual steps are carried out.

Alternatively, you can delegate full responsibility for implementation by stating the goal without specifying the concrete steps.

Example
Develop a schedule with specific content for executive coaching. The content should consist of individual and group exercises as well as instructional units.

In our example, the coaching should build on an existing training program. To provide this information, upload the existing workshop concept as a file and simply extend the prompt to reference it.

Example
Attached you will find an existing workshop concept. Plan the new workshop based on this concept. Adjust exercises and content where necessary or appropriate, and briefly justify each of your adjustments.

This saves you a lot of time, as you do not have to explain your existing knowledge in a lengthy and complicated way or have the results adjusted in multiple steps.

Depending on the model, you can also upload and analyze images, such as photographs used as workshop records.

3.7.3 Specification of the Final Result

In the third part (*Result*), you specify exactly what you expect the result to look like. This includes both format and length as well as content. For example, you can require that a list of a certain length be created, or that formulations be written in a particular style.

You can also refer to result formats already known to the LLM. This could be something the model is already familiar with, such as the structure of a marketing plan or a summary. Alternatively, you can provide or describe the format as an example. Finally, you can also attach example files and refer to them.

In our example, for the Result part of the prompt, you simply refer to the files with the existing workshop structure that have already been attached. In the previous section, the LLM was already instructed to follow the content, so in this step you only refer to the workshop format.

Example
The result should be structured in the same way as the schedule in the file "Schedule.pdf".

Alternatively, you can define the result more specifically or provide a textual example. For instance, use the following template:

Example
The schedule should consist of individual blocks, each structured as follows:

###

Start time – End time: Title

Number of participants

Description

###

Since you do not specify exactly what each component means, the language model will make assumptions and, for example, replace *Start time* or *Title* with specific information, but may begin the description with "Description:".

3.7.4 Definition of an Example

In addition to these first three steps, this framework also includes at least one concrete example (Example) to guide the LLM. Again, you can either refer to something the model already knows or provide specific examples in the prompt.

If you want to have a social media marketing campaign developed, you can refer to other companies that have run successful campaigns. If you want to have an offer drafted, you can, similar to the previous section, provide a sample structure in the prompt.

In the current example, you can specifically refer to the already existing concept. It may be necessary to explicitly state that the result should be adapted in terms of content.

Example
You can find an example of a well-planned and structured workshop in the file "WorkshopStructure.pdf".

3.8 The R-I-S-E Framework (Role-Input-Steps-Expectations)

If you want to develop a plan to achieve a clearly defined goal for a specific situation for which you have already collected data, the R-I-S-E Framework (Role, Input, Steps, Expectations) may be the right approach. You begin by describing the role from whose perspective the plan should be developed. Next, you explain the available data about your situation and ask for the necessary steps to achieve the goal. This goal is then defined in the final part of the prompt.

3.8.1 The Role in the Prompt

The first step is to define the role to be assumed in the prompt. In general, it is sufficient to simply state the role, especially if it is a common one, such as a recognized job title. Let us use the example of developing an engagement plan for project stakeholders. The LLM should therefore assume the role of a project manager.

Example
Act as an experienced project manager.

If more information is relevant, it should be defined directly with the role. You can refine the role itself with details, for example by specifying particular expectations or perspectives. The context in which the role is to be assumed can also significantly influence the behavior and thus the results of the prompt.

Example
You are an experienced project manager of a project being conducted according to the waterfall model. It is a large and critical project, and you have overall responsibility.

3.8.2 Explaining the Data and Information for the Prompt

In the second part (Input), you explain which data you are providing to the prompt. For example, you can reference examples in text form within the prompt. Often, it is more practical to provide the data as files, such as PDFs or images. These can then be briefly referenced in this part of the prompt. This not only keeps the prompt shorter, but also leverages the language model's ability to understand attached files. Explain the attachments as needed.

Example
Attached you will find an overview of all project stakeholders, including their expectations and experience. You will also receive a risk matrix for the project, which includes the stakeholders.

3.8.3 Requesting Steps Toward a Solution

After the previous two components have provided a comprehensive description of the situation, the third part asks the LLM to develop concrete steps to achieve the goal. The goal has not yet been further defined and can therefore be succinctly described here.

Example
Develop a step-by-step plan for stakeholder management for the project.

Both a multi-step plan and, as in this example, stakeholder management are broad subject areas. It is therefore advisable to further specify your requirements. You can describe which steps you expect in the result or which you consider essential. Depending on the complexity, it may also make sense to break steps down into sub-steps or to request additional explanations. Take advantage of the extensive options described in Sect. 2.4.

In the example prompt, the goal was only briefly outlined. Here, too, it can be helpful to provide further information and boundary conditions. Focus here solely on your expectations for the outcome of the prompt, i.e., the plan to be developed.

Example
Develop a step-by-step plan for stakeholder management for the project. For each step, specify the objective and the required approach. Explain why each step is useful and what benefits it brings.

3.8.4 Defining Target Expectations

The final part of the prompt defines the expectations for the outcome. These expectations relate to the situation that will be achieved by implementing the steps of the plan. In other words, you describe the ideal target situation. Inform the model of your expectations for this. The more specifically you define what you want to achieve with the plan, the better the plan will be tailored to your needs. Here, too, you can use

concrete examples, for instance by attaching files or embedding relevant text. You can also refer to generally known examples that are part of the language model's knowledge base. Finally, you can specify concrete target values to be achieved.

In the example, improved stakeholder satisfaction is defined as the goal. The existing key figures are briefly referenced, and the specific target value is then set.

Example

The goal is to optimize stakeholder satisfaction. It is currently at 67% and is determined by weekly surveys. An example of such a survey can be found in the file "Stakeholder-Quest.pdf." Within the next 90 days, satisfaction should be increased to 85%.

3.9 The Expert Prompt Framework (EPF)

The following section introduces the Expert Prompt Format. It combines many of the tips discussed so far into a format that is easy to apply and, after a few uses, easy to remember.

3.9.1 Overview

The Expert Prompt Framework (EPF) is a framework that describes the task for the language model in several steps, in a very precise and structured manner. Like various other frameworks, it uses roles to outline the basic context of the results. In several steps, the expected outcome and the activities required to achieve it are then specified. Unlike other frameworks, it also explicitly names exceptions and undesired results. Finally, it defines the output format, the target audience, and clarifies the author's expectations with one or more examples.

Example

Act as a [expert role].

I need [description of the result].

You will [precise task description].

In doing so, you will [further details].

Please [list exceptions].

Develop the result for [target audience].

Deliver the final result as [format].

Here is an example: [example].

The format is quite comprehensive, and writing prompts using this framework does require some time. However, the results will be much better aligned with your expectations and requirements. Manual post-processing or further iterations with the language model are often unnecessary.

Such a comprehensive prompt framework also has another important effect: as the author, you engage more deeply with the question and your own expectations. It is not uncommon for new perspectives to emerge and for the question to evolve as you write. In meta-prompting, these details are queried directly by the LLM (see Chap. 5).

The wording of the individual components is generally used as shown, but can of course be varied linguistically. Ultimately, it is important that the LLM can understand and correctly implement the individual components.

3.9.2 The Expert Role

As with various other frameworks, the first step is to define as concretely as possible the role that the model is to assume. The role influences how the language model assesses the relevance of information, chooses its tone, and fundamentally determines the format in which results are developed.

If the chosen role is not specific enough or additional context is required, further information can be provided. The more precisely the role is defined, the better the results will be in the end.

Example
Act as an online marketing expert.

Or more specifically:

Example
You are an expert in SEO optimization with a focus on website performance and user experience (UX). Your experience centers on CMS-based websites for medium-sized companies.

3.9.3 Description of the Result

The next step is to describe the desired result. In practice, this is usually quite brief, as the steps to achieve this result are described in the next component of the framework. The aim here is to formulate the target outcome as precisely as possible. This may mean that a short sentence is sufficient to clearly state the desired result. On the other hand, it can also be appropriate and important to provide additional details. This allows the result to be specified in advance and provides a meaningful framework.

Example
I need an article about lawn care.

Or more specifically:

Example
I need the text for a short newspaper article on the topic of professional lawn care.

It is important not to take over the tasks of the following parts of the framework. The result description should not include the process for achieving the result or consist solely of it. In practice, it is usually quite easy to communicate the desired result precisely.

The challenge often lies in refraining from specifying requirements for format and structure at this stage and instead incorporating them

later. The boundary here is fluid, and in the end, the result is more important than a well-phrased prompt. Nevertheless, it is helpful to specify format and structure separately.

3.9.4 Task Description

In the third part, the specific activity is described in more detail, outlining the exact steps to be taken to achieve the previously defined result. Depending on the complexity of the overall question, this section may consist of just a few keywords or a detailed breakdown of individual steps.

Not every possible variation is explained, nor are all details of the process comprehensively laid out. Rather, this section provides a general outline of the expected approach. Additional information for elaboration is provided in the next step of the framework.

Example
To do this, you will write an article on the topic from your perspective as an expert.

This separation of the description of the approach into this and the following step is comparable to the concept of OKR (Objectives and Key Results). In other words, the "big picture" of the expected approach of the LLM is explained. Further key steps are described in the next step.

3.9.5 Further Details on Implementation

The following section further specifies the approach for achieving the overall prompt result. Additional details are added here that can further clarify both the process and the result. This may involve specifying a particular method or sequence of steps. This is also the appropriate place to prescribe specific frameworks for the approach or to define exceptions and alternatives.

Example
You will use the AIDA framework for storytelling and write at least two paragraphs for each component of the framework. Develop an engaging storyline for this purpose.

If interactive engagement with the model is required during processing, this can also be specified at this point.

Example
First, describe the storyline and ask me what adjustments I need. Incorporate these into the storyline. Ask further questions if there are unresolved points or if you have additional useful ideas. Only write the article after that.

3.9.6 Exceptions

Just as it is useful to concretely describe the expectations to be implemented, it can also be helpful to specify approaches or results that the model should avoid. This can eliminate common sources of error and prevent typical, undesired behavior of language models in advance.

The EPF uses the word "Please" followed by a list of exceptions in any form, such as individual sentences or bullet points. Other phrases are also possible, such as "Note the following exceptions:…" or "Make sure that…".

Exceptions can relate to any part of the prompt. For example, the expert role can be restricted here if necessary. The overall result of the prompt can also be subject to exceptions, as can the approach described in the previous two sections. Exceptions can also refer to upcoming prompt components such as the target audience and format.

Example
Please do not use technical jargon or colloquial language. Please do not use emojis.

3.9.7 Target Audience of the Prompt

In addition to the expert role, the target audience is an important lever for influencing the result. The same principles apply here as for the expert role in the prompt: the more specifically the target audience is defined, the better the result will fit in the end. While in some scenarios simply naming the target audience is sufficient, in others it may be necessary to provide more context.

Specifying the target audience can have a significant impact on the generated text. Depending on the audience, the LLM will use specific expressions and technical terms, or automatically explain terms that cannot be assumed to be known by the audience. The language of an entire target group can also be imitated. For example, consider current youth language. Regional target audiences can also be defined, allowing the LLM to incorporate local language nuances or dialects. Where appropriate, suitable emojis or hashtags—common in social media for marking specific topics or people, indicated by a preceding "#"—can also be included in the text.

If the target audience is to be described in great detail, it may be useful to use personas. These are idealized profiles representing typical members of the target audience. Since personas can be quite extensive, it may be advisable to attach them as files and reference them, rather than trying to include all the information in the prompt itself.

Example

The article should be written for middle-aged homeowners with a garden. You will receive a persona of an ideal reader as a PDF file.

3.9.8 The Format of the Result

The next step is to define the format in which the result should be presented. Depending on the model, it may be possible to generate files directly, such as Microsoft Word documents. Regardless, it can be specified, for example, that a list or a suitably formatted text should be produced. The formatting options from Sect. 2.4 apply here.

Example
Write the article with 2000 words and format it into meaningful paragraphs. Bold the most important phrase in each paragraph. Additionally, write a brief introduction and a short conclusion.

3.9.9 Providing an Example

Finally, one or more examples are provided in the prompt. This part can be very useful and effective when the requirements for the output are quite extensive or complex and a detailed target format description would be necessary. If the objective and result have already been described effectively enough using the other components of the expert prompt framework, it is often simpler to omit the example at first.

Example
Attached you will find last year's article on a different topic for your reference.

3.10 At a Glance

- Prompt frameworks provide clear, traceable structural guidelines for creating effective prompts.
- There are specific frameworks for different objectives.
- Prompt frameworks can be combined with additional prompting techniques and thereby refined.
- It has been shown that for many prompts, it is beneficial to assign the LLM a specific role in which it should act. This focuses the system on a particular task and results in more concrete answers.
- When the model knows who the intended audience of the text is, the generated text can be much better—and, above all, automatically—tailored to the respective expectations and requirements.
- For more complex queries, it has proven successful to use a deductive approach and work step by step with the model to reach the result.
- The expert prompt framework provides an easy-to-use blueprint for most queries.

4

Advanced Techniques for Prompts

Abstract This chapter presents advanced techniques for creating effective prompts. It first describes the so-called zero-shot, one-shot, and few-shot prompts, which guide the LLM toward solving the actual problem by providing varying numbers of examples. The cognitive verifier pattern is then explained, in which the LLM independently queries essential information required to solve the task. This is followed by chain-of-thought prompts, where the solution is developed step by step together with the LLM, and the multi-role analysis, in which the LLM assumes different roles and thus approaches the problem from various perspectives.

Frameworks can be used to structure prompts in a way that already yields significantly better results with LLMs. In addition, there are advanced techniques that can further improve quality considerably. For example, problems can be broken down into subproblems or answered using different roles or contexts. All of this is handled entirely by the language model, without any manual intervention. Moreover, these techniques can also be combined with prompt frameworks such as those described in Chap. 3.

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_4

4.1 Zero-Shot, One-Shot, and Few-Shot Prompts

To make problem-solving easier for an LLM, examples can be included in the prompt. This clarifies how to handle specific situations and what your expectations or understanding are in each case. Depending on the number of examples provided, a distinction is made between zero-shot, one-shot, and few-shot prompts.

4.1.1 Zero-Shot Prompts

For simple questions, it is sufficient to submit a single, specific request to the LLM to obtain a good result. It is not necessary to include examples in the prompt. These single requests are called zero-shot prompts. Here are two examples of zero-shot prompts.

Example 1:

Example
Who invented penicillin?

Example 2:

Example
Determine the tone of the request.

Request: Why are you not responding to me?

While the first example is easy to answer based on facts, the second may require more context. The tone of the request is determined by the model based on its knowledge, without knowing your specific context. To provide this context, a detailed role description could be included, for example by using the R-T-F framework (see Sect. 3.4). Alternatively, the prompt can be enriched with relevant examples, turning it into a one-shot or few-shot prompt.

4.1.2 One-Shot Prompts

If simple, direct prompts do not yield the desired result, a specific example can be included in the prompt to clarify the approach or your expectations. This is usually done with section markers. In addition, the task can be further specified before or after the example.

Here is the example for determining tone as a one-shot prompt.

Example

Determine the tone of the request. Here is an example:

Request: Can you help me?

Tone: Friendly

Request: Why are you not responding to me?

For this task, an example is first constructed using sections, and the pattern is then continued, but the answer is omitted. The LLM will generate a response following the format of the example. This response is typically better than with a zero-shot prompt, as the example also implicitly communicates the expected output format. The format is important: the prompt essentially simulates a request and a response, as the language model itself might generate. There is no further explanation as to why the result is as it is; instead, the model is trained with the example from the prompt and applies these insights to the task at hand.

4.1.3 Few-Shot Prompts

For more complex problems, it can be useful to include several examples. This clarifies the approach to the solution, and the LLM can generate more specific results. The examples simulate requests and results and are used to train the model in this context. The model will then draw conclusions from them and apply these to the task. If a prompt contains several such examples, it is called a few-shot prompt.

Here is an example of a few-shot prompt:

Example

Determine the tone of the request. Here are a few examples:

Request: Can you help me?

Tone: Friendly

Request: Help me!

Tone: Demanding

Request: I find it rude that I have not received a response.

Tone: Unfriendly

Request: Why are you not responding to me?

The requests and their results are not explained in further detail. Instead, the language model is prompted to draw its own conclusions from the examples and apply them to the task. In doing so, it takes into account both the context of the examples and the format.

4.2 The Cognitive Verifier Pattern

To obtain good answers for complex questions, it may be necessary to include a large amount of detailed information in the prompt. While various prompt frameworks already make this easier, there is a fundamental challenge: you need to know which information is relevant and to what extent. This task can also be delegated to the LLM. With the cognitive verifier pattern, the model is instructed to request relevant information needed to process the prompt.

The structure is very straightforward. At the beginning of the prompt, an instruction is included to break down the following task into three questions, have these questions answered, and then use the results to process the prompt.

Example

Generate three questions for the following task that are useful and necessary for effective completion. Have me answer these questions one after the other and use my answers to complete the task.

The actual prompt then follows. The model will, according to the instructions, ask three questions, have them answered, and then develop the result for the prompt. The number of questions is generally variable. More than three are often time-consuming to answer, and fewer may omit important aspects. If it is already clear that three questions are not suitable, a different number can certainly be chosen.

Of course, you can also let the language model determine the number of questions. To do this, instead of specifying a concrete number, rephrase the prompt so that a reasonable number of questions should be asked. However, it may be useful to specify a range (minimum to maximum) to ensure a certain level of quality while not overshooting the mark.

4.3 Chain-of-Thought Prompts

Many prompts require going through several steps to generate the desired result. If the outcome does not fully meet the requirements or significantly deviates from expectations, it may be that the LLM has gone astray during processing. Instead of extensively reworking the prompt and adding large amounts of information, it is often sufficient to have the model present and, if necessary, explain the individual steps in its reasoning. This makes the chain of thoughts and assumptions (*Chain of thought*) transparent.

To do this, simply add a short phrase at the end of the prompt instructing the LLM to justify its approach and outline the steps taken. After you have described the task in the prompt, include the following addition:

Example
Proceed step by step.

This ensures that each sub-step and its respective result are output individually. Just as some people find it easier to arrive at solutions by thinking out loud, this prompt extension often leads to improved results. In addition, you gain the ability to trace the model's reasoning. This allows

you, for example, to have individual sub-steps adjusted afterwards, thereby further improving the quality.

> Sometimes, this phrase is interpreted as a formatting instruction, resulting in the output being structured as individual steps—without actually changing the approach. In such cases, rephrase the prompt to make it clear that intermediate steps should be explicitly stated.

This type of chain-of-thought (COT) prompt is referred to as a *Zero Shot COT Prompt*. It prompts language models to present and explain their reasoning process. However, you do not provide examples in the prompt. The principles of few-shot prompts (see Sect. 4.1.3) can also be applied to chain-of-thought prompts. In this case, the prompt includes concrete examples, each consisting of a question and an answer. Unlike classic few-shot prompts, the reasoning process is described in addition to the solution. This enables the LLM to follow your approach and apply it to the final question. The introductory phrase that triggers the reasoning in a zero-shot COT prompt is omitted when examples are provided. If the generated solution does not meet your expectations, you can still use this phrase to elicit more details about the reasoning and thus improve the result.

The following example illustrates the principle. The model is tasked with calculating the production capacity of a machine. For this, some models, such as GPT-4o from OpenAI, often generate a Python script (a common programming language) to perform and execute the calculation. However, you usually do not see this intermediate step, as it is carried out automatically in the background. The way the prompt is formulated causes the LLM to perform the calculation differently. Due to the low complexity of the task, the results are also correct. Following the pattern, the model will calculate and round the result correctly. However, the results may vary depending on the version. Try the prompt, for example, with GPT-3.5, GPT-4, and GPT-4o from ChatGPT, or with any other LLM to observe the differences.

Example

Question: A machine produces 22 units per hour. Two additional machines are purchased, each with half the production capacity. One of the new machines operates at 150% capacity. How many units are produced per hour?

Answer: 22 units per hour + 11 units per hour + 11 * 1.5 units per hour = 49.5 units. Since partial units cannot be produced, the result is rounded down. 49 units are produced per hour.

Question: A machine produces 10 units per hour. Two additional machines are purchased, each with double the production capacity. One of the new machines operates at 9% capacity. How many units are produced per hour?

Answer: 10 units per hour + 20 units per hour + 20 * 0.09 units per hour = 31.8 units. Since fractional units cannot be produced, the result is rounded down. 31 units are produced per hour.

Question: A machine produces 7 units per hour. Two machines are purchased, each with 75% of the production capacity. How many units are produced per hour?

4.4 Multi-Role Analysis

In a business context, many tasks inevitably involve multiple stakeholders. Each of these stakeholders brings their own experiences, expectations, and preferences—that is, their own context—and may identify different opportunities and challenges in a given situation.

This fact can also be leveraged with an LLM. By instructing the model to assume different roles and address the task from each perspective, you can gain insights that might otherwise be easily overlooked.

Here is an example of developing requirements for a policy on working with artificial intelligence in companies. The model is to independently identify relevant stakeholders, assume their roles, and consolidate the results at the end.

Example

Requirements for a policy on the use of artificial intelligence are to be developed. The task should be addressed from three different perspectives.

Identify and name the three most important stakeholders in a company who would be affected by such a policy.

Develop requirements for the policy from the perspective of each of the three stakeholders. Present the requirements as a simple list of five items each.

Then indicate where there are synergies and potential conflicts between the three stakeholders.

Depending on the task, working with multiple roles can significantly increase the effort involved. It is particularly suitable for tasks where concise results can be produced, such as assessing situations or developing concepts. In some cases, this multi-role approach can also be effectively combined with other prompt frameworks.

4.5 At a Glance

- With zero-shot, one-shot, and few-shot prompts, you can provide the LLM with examples of possible solutions to a task. The system learns how to respond and seeks a similar solution.
- The cognitive verifier pattern enables the system to independently ask solution-relevant questions without further instructions, thereby improving the quality of the result.
- With so-called chain-of-thought prompts, the LLM works step by step from the problem statement through several intermediate steps to the final result. The system's individual "thought steps" are made transparent and can be questioned, adjusted, or expanded.
- With multi-role analysis, the LLM can adopt different perspectives. This provides various solution approaches and can be very useful, as a given problem is examined from multiple angles. This often brings to light new insights that might otherwise have been overlooked.

5

Metaprompting

Abstract This chapter introduces the concept of metaprompting. In this approach, the LLM is not used directly to answer a given question; instead, the system is first employed to generate an appropriate prompt for creating the subsequent solution. Prompts can be generated either for the LLM itself or for third-party systems. For example, ChatGPT can generate prompts for creating images using Midjourney. Additionally, metaprompting can be used to further optimize existing prompts. Finally, the Valisory Prompt Studio is presented as a complex, fully automated prompt that integrates all previously described concepts into a comprehensive workflow.

Generative language models are frequently used to produce text that is subsequently utilized by humans or other systems and applications. The techniques presented in Chaps. 3 and 4 are employed to make such requests more effective. A valuable capability of modern language models, however, is their ability to generate prompts themselves. Moreover, they can also compose effective prompts for other generative AI systems. For example, ChatGPT can be used interactively to develop a high-quality prompt for the image AI Midjourney, which can then be

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_5

used to create images. The techniques of metaprompting are explained in this chapter.

5.1 Fundamentals and Principles of Metaprompting

As discussed in Sect. 2.1, communication is inherently a complex endeavor. To reduce this complexity when working with language models, various prompt frameworks have been introduced. Another, and combinable, approach is **metaprompting**. In this approach, a language model is used to write a prompt for another model or for itself. The term derives from the fact that this method operates on a meta-level: instead of writing the actual prompt, a prompt is developed for generating another prompt. Metaprompting is often an iterative process in which an optimal prompt is developed step by step through interactive refinement.

It is considered meta-level because, in the classical approach to prompt engineering, the focus is on constructing the request to the LLM in such a way that the result is as good as possible. In metaprompting, the prompt engineering effort is now directed at designing a prompt so effectively that the language model can use it to develop a prompt for another model, which is then used in that model to generate the actual result. In principle, this concept could also be applied recursively to metaprompting itself, by using a language model to write a prompt that is used to write another prompt, which in turn generates a prompt that produces the final result.

But why go to the trouble of working "indirectly" with a generative AI system? On the one hand, for sufficiently complex tasks, it can be highly beneficial to invest time in developing a prompt in order to minimize the need for post-editing the final result. Instead of doing this alone, one can leverage the capabilities of an LLM to develop the prompt. On the other hand, there are indeed differences in prompting for various AI models—at the very least when comparing systems for generating text, videos, or images. With the help of metaprompting,

effective prompts can be developed for these scenarios, ensuring that all necessary information is included and common sources of error are avoided.

5.2 Approaches to Metaprompting

The following sections first introduce various approaches to metaprompting. Finally, the "Prompt Studio" is presented as a solution that can automatically generate optimal prompts for any task.

5.2.1 Self-Referential Prompts

The fundamental approach of metaprompting is self-referentiality. Here, the language model is prompted to analyze, reflect on, and identify potential improvements in a prompt for itself—essentially creating a feedback loop.

To do this, the model is first presented with a prompt, which it is to analyze and evaluate. The prompt is marked with quotation marks or as a text embedding. To effectively further develop the prompt, it is helpful, for example, to have the model generate a numbered list of suggestions. This makes it easier to refer to specific ideas, further develop them if necessary, and incorporate them into the prompt.

Example

You will receive a prompt below. Analyze and evaluate whether this prompt is precise enough to achieve optimal results. Provide concrete suggestions for improvement. Use a numbered list for each.

The prompt:

###

Explain the core function of a neural network in simple terms.

###

The LLM will then analyze the objective of the prompt and immediately suggest possible improvements. In some cases, it may also directly produce a better version. The results can then be used to iteratively continue working with the language model in a dialogue until the desired final version of the prompt is achieved.

You can further adapt the above prompt to encourage iterative and dialogic work with the language model on the actual prompt. To do this, include an instruction for the model to ask you questions and incorporate your answers into the prompt. The model will often ask questions that you may not have originally considered.

Example

Develop up to three questions. Present them to me in a numbered list and incorporate my answers into the prompt.

Another minor optimization is to always display the current version of the prompt. This has two advantages: First, you always know what the current result looks like and can provide more effective feedback. Second, it helps address the limitations of the context window. Therefore, it is practical to always keep an up-to-date version. To keep the process as structured as possible, it helps to define a format—for example, first output the current prompt, then possible suggestions for improvement, and finally the list of questions. This ensures that each step follows the same format and adequately addresses the inherent variability of an LLM's responses.

To avoid having to laboriously copy the actual prompt from the overall response at the end, simply instruct the language model to provide you with just the prompt.

Occasionally, the language model will independently execute the prompt at a certain point—even if you have explicitly instructed it not to. So far, no fully reliable solution to this issue has been found.

5.2.2 Iterative Optimization through Dialogic Development

Similar to the approach of self-referential optimization (see Sect. 5.2.1), prompts can also be developed or further refined from scratch. To do this, the model is given the objective, optionally presented with a prompt, and then improvements are made iteratively and dialogically. If a prompt is being developed for another AI model, it is important to specify this and, if relevant, to indicate versions or specific requirements.

The tips from Sect. 5.2.1 regarding format and interaction also apply here. Especially when many (expected) iterations are involved, it can be helpful to define an output format. Furthermore, it is highly beneficial to have the LLM independently identify questions or comments.

Example

You are a prompt development assistant. Our goal is to create a prompt for Google's Gemini that generates a precise checklist for a software architecture review. The prompt should be in English.

We will develop the prompt step by step. In each step, you will first output the current version of the prompt. Then, provide a bullet-point list of possible improvements. After that, present a numbered list of questions relevant to the development of the prompt, with a maximum of three questions. Incorporate my answers into the next version of the prompt.

At no point should you execute or answer the actual prompt yourself.

In this example, Google's Gemini was defined as the target model. The language of the target prompt was also specified as English. Role prompting (see Sect. 3.2) was also used. The second part defined the procedure, particularly the format of the interactions. Finally, the model was instructed not to execute the prompt itself (see Sect. 5.2.1).

5.2.3 Evaluation and Development through Role-Based Abstraction

This type of metaprompting is a variation of iterative optimization (see Sect. 5.2.2). Here, the model is prompted to assume a role that abstracts the development process (e.g., moderator, critic, or advisor). The prompt is then further developed within this role.

Example

You are a critical prompt reviewer. Analyze the following prompt: "Generate a list of marketing ideas for a product launch event."

1. Evaluate whether the prompt is specific enough to yield precise results.
2. Suggest ways in which the prompt could be improved.

The idea is that such an abstracted role is more likely to assess and optimize criteria such as clarity, accuracy, and specificity. The focus here is less on the creative (further) development of a prompt.

For extensive prompt development, it can be very helpful to include such a critical reviewer in the development process, for example by having the prompt outcome evaluated by a role prompt at the very end. Furthermore, it is conceivable to combine several language models, for instance, developing a prompt for Google's Gemini in OpenAI's ChatGPT and then having it critically quality-assured in Anthropic's Claude.

5.3 The Valisory Prompt Studio

Using metaprompting techniques, we have designed a comprehensive metaprompt that enables the step-by-step development of high-quality prompts: the **Valisory Prompt Studio**. To achieve this, we applied the techniques presented in this chapter and iteratively optimized them through testing in line with prompt engineering best practices.

The Valisory Prompt Studio can be used to develop entirely new prompts or to improve existing ones. It employs an iterative process in which improvements are identified in dialogue with the language model and incorporated into the prompt. The Prompt Studio can be implemented as a predefined assistant or chat (in ChatGPT, "CustomGPT") and made available to employees.

The prompt for the Valisory Prompt Studio is structured as follows: first, basic rules and assumptions for the entire interaction are defined. This includes the objective and a role, as well as language and tone. Next, the general procedure is described, independent of the specific scenario. This is followed by a description of the steps to be taken if a prompt is to be improved. Finally, the largest section defines how the development of a prompt should be carried out.

5.3.1 The Basic Rules of the Valisory Prompt Studio

The first section of the prompt defines the fundamental approach and the framework conditions. First, the Valisory Prompt Studio is assigned the role of an "experienced prompt engineer specializing in developing effective prompts for language models" (text from prompt). This role prompting (see Sect. 3.2) improves the process, assumptions, and quality of the development. At the beginning, the model is also informed of its name so that it can meaningfully communicate its actions to the user.

Second, the objective and the two possible scenarios are explained. The goal of the Valisory Prompt Studio is to support a user in creating the "best possible" prompt. To this end, either a completely new prompt can be developed or an existing one can be improved.

Example

You are the Valisory Prompt Studio, an experienced prompt engineer specializing in developing effective prompts for language models. You support the user in developing the best possible prompt for their needs. The user can either develop a completely new prompt or improve an existing one.

> The following describes the procedure, which you must not deviate from.
>
> The process is fundamentally iterative: through questions and feedback, the prompt is further developed step by step.
>
> At no point do you execute or answer the developed prompt yourself.
>
> You communicate in a formal and polite manner. The language of interaction with the user is the respective national language (unless otherwise specified). The language of the prompt to be created is also the national language, unless the user explicitly requests otherwise or provides a prompt for improvement that is written in another language.
>
> The following description is divided into sections (A, B, and C). Start with section A.

The procedure is then described. At the outset, it is explicitly stated that this procedure should not be deviated from. Of course, this requirement is somewhat vague, as the procedure described below is not comprehensive in many places, and the LLM will bring its own interpretation. Nevertheless, this note helps ensure that the model largely adheres to the plan and format of the prompt.

The language model is then given an outline of the basic approach: it is iterative, questions are asked and answers are given, with which the prompt is further developed. Phrases such as "iterative" and "step by step" signal to the model that a result is not expected immediately after a single step. This is intended to achieve better results from the model in terms of prompt chaining (see Sect. 3.1).

The model is also instructed not to answer or execute the developed prompt itself. Without this instruction, after several iterations, the process of improving the prompt may become mixed with answering it. This makes the chat with the LLM confusing, and the actual improvement can no longer be meaningfully continued. In our tests, including this phrase prevented exactly this situation, although it can still occasionally occur.

Next, there are notes regarding language and tone. The model should be formal and polite. Interaction with the user should (initially) take place in the respective national language. The prompt to be developed should also be written in the respective language, unless the user requests otherwise or provides a prompt to be improved that uses another language.

Finally, the structure of the actual procedure is defined. It is divided into three sections: Section A defines prerequisites; Section B describes the procedure when a prompt is to be improved; and Section C addresses the case where a new prompt is to be developed.

5.3.2 Section A: Prerequisites

The first block of the procedure description defines the prerequisites that the model must establish so that a new prompt can be developed or an existing prompt improved. The section is marked with the section label "A".

First, the language model for which the prompt is to be developed or improved must be defined. The phrase "first of all" is important here—otherwise, the Valisory Prompt Studio may ask for this information later or not at all. Second, it is ensured that the objective of the interaction is clear; either the development of a new prompt or the improvement of an existing one. For both situations, exceptions are built in so that the question does not need to be asked if the user has already provided the information.

These two queries are intended as support for the user to improve usability. Since interaction with the language model is interactive, such information can be provided at any point along the way. The Valisory Prompt Studio is to explicitly ask for this information to support the user and ensure that nothing is forgotten.

Example

A: Your general procedure

- If the user has not specified it, first ask for the language model for which the prompt is to be developed.

- If the user has not specified it, secondly ask whether a new prompt is to be developed (continue with B) or an existing prompt is to be improved (continue with C). Then proceed with the respective section below.

5.3.3 Section B: Improving a Prompt

The second block of the procedure description addresses the case where an existing prompt is to be improved. In the end, the process is quite extensive: the prompt to be improved may still need to be provided by the user. Then, an analysis must be conducted and optimizations made.

In detail, however, the procedure is fundamentally very similar to the other possible situation, where a new prompt is to be developed. The only difference is the starting point: in one case, a prompt already exists, in the other, it does not. If we assume that optimizing a user's existing prompt is equivalent to optimizing a prompt that was drafted in a first version by the language model, the subsequent steps only need to be defined once for both situations. Thus, the entry points are defined differently, and part of the prompt is reused.

Example

B: In the case that a prompt is to be improved:

1. Ask for the prompt to be improved if the user has not yet provided it.
2. Analyze the prompt and identify the objective. Ask the user whether the objective you have determined is correct or if the user would like to define a different objective.
3. Continue with step 3 of Section C.

If, at the beginning of the interaction with the Valisory Prompt Studio, the user has indicated that a prompt is to be improved but has not yet provided it, the prompt explicitly defines that the user should be asked for the prompt.

Before the improvement begins, the Valisory Prompt Studio should next derive the objective from the given prompt and have the user confirm it. The user is also given the opportunity to adjust the objective. This ensures that there is congruence between the prompt and the objective, or that any lack of congruence can be corrected by the Valisory Prompt Studio.

Afterwards, the Valisory Prompt Studio should proceed to Section C and continue with step 3 there.

5.3.4 Section C: Developing a New Prompt

This final block describes the remaining two building blocks: first, the development of an entirely new prompt, and second, the iterative process for optimizing the prompt.

In the first step, the model asks about the objective to be achieved with the prompt. Here, the model is explicitly allowed to ask follow-up questions if the answers are "unclear or not precise" (text from prompt)—the exact interpretation of this is left to the language model's discretion.

In the second step, based on the defined objective, an initial version of the prompt is developed. The LLM refers to the initially specified target language model, any required language changes, and any further adjustments. To keep the model's output concise, it is instructed not to output this first prompt version, but to proceed directly to step 3.

Example

C: In the case that a prompt is to be developed:

1. Ask about the objective of the prompt. Ask follow-up questions if the user's answers are unclear or not precise.
2. Develop an initial version of the prompt. Do not output this version, but proceed directly to step 3.
3. Analyze the current prompt from two perspectives:
   a. As a critical prompt reviewer
   b. As a creative prompt developer

4. Always output the following:
   - The objective of the prompt
   - The current prompt
   - Suggestions for improvement from the perspective of the critical prompt reviewer
   - Relevant questions (maximum 3) for improving the prompt from the perspective of the prompt developer
5. The user will answer the questions. Ask follow-up questions if the answers are unclear, imprecise, or not meaningful.
6. Further develop the prompt using the answers to the questions and any additional comments from the user.
7. Then return to step 3 in this section. Continue this iterative process until the user is satisfied.

Step 3 is also the step to which the Valisory Prompt Studio jumps after an existing prompt has been specified for optimization. From this step onward, the actual optimization work on the prompt begins. Whether the prompt originally comes from the user or was developed by the model is irrelevant.

In step 3, the prompt is analyzed. Two roles are used for this purpose. The first is a "critical prompt reviewer" (see Sect. 5.3.2). The idea is to identify opportunities and challenges with as objective a focus as possible. A reviewer was deliberately chosen—a role whose strengths lie in evaluation rather than (further) development. In contrast, the second role is the "creative prompt developer," who is intended to find opportunities for further development using a creative approach.

Next, in step 4, the Valisory Prompt Studio should output the current status. The format is always the same. First, the defined objective is stated, followed by the current version of the prompt. This allows the user to easily check whether the prompt is (still or already) fundamentally suitable for achieving the objective and, if necessary, whether it might already be sufficiently good. In addition, the user should have to remember as little as possible.

Comments on the prompt are then output. These are based on the critical review from step 3. This is followed by up to three questions about the prompt, the answers to which can help improve it. These

questions were developed by the second role in step 3. Within step 4, reference is always made to the roles from step 3.

In the fifth step, the user should answer the questions, and the model should incorporate the answers into further development. In fact, the LLM would usually behave this way even if not explicitly instructed. By specifying this behavior, however, it can be better controlled. At this point in the process, control is handed over to the user, who can answer the questions.

Afterwards, in step 6, the language model should further develop the prompt using the answers. Although it was already mentioned in the prompt that the user's answers should be incorporated into the development—and this would indeed be the natural behavior of the Valisory Prompt Studio—making this explicit does improve the quality of the implementation.

> In fact, it has been shown that repeatedly stating or describing a task leads the model to adhere to it more closely. This technique is used, for example, in the safeguarding of AI agents to prevent undesired behavior.

The final step of the prompt then defines the iterative process. The model should return to step 3 and continue from there. This process should be repeated until the user is satisfied. This condition is important so that the model does not independently determine the end and interrupt the process. When the user is satisfied is not defined, so the language model may subtly inquire or point out satisfaction at various points. In practice, however, this is not relevant, as a chat with an LLM is usually not formally ended; instead, the results are used elsewhere upon completion without further feedback.

5.3.5 Provision of the Valisory Prompt Studio for Users

The Valisory Prompt Studio can be used in various ways. The seemingly simplest case is to insert the prompt into a new chat. Interaction with the Valisory Prompt Studio can then proceed as described. However,

there are two disadvantages: first, the prompt is quite extensive and therefore often impractical. Second, it can be assumed that the prompt will be further developed over time and would then need to be redistributed—especially with multiple potential users within a company, this can be a considerable logistical challenge. It therefore makes sense to provide the Valisory Prompt Studio as an AI agent, for example, as a CustomGPT in OpenAI's ChatGPT.

Central provision as an agent relieves users of the need to store the prompt digitally somewhere and copy it into new chats each time. On the other hand, the company benefits from the prompt being stored centrally and further developed there—completely transparently for users.

In conclusion, it should be noted that the Valisory Prompt Studio should not be used for every request to an LLM. That would often be the proverbial "using a sledgehammer to crack a nut." Rather, the Valisory Prompt Studio should be used when comprehensive or highly specialized prompts need to be created or optimized. For many requests, the frameworks described above are entirely sufficient. Nevertheless, the Valisory Prompt Studio is a very powerful tool and demonstrates that, with the help of prompts, even highly complex tasks can be solved in a very structured and transparent manner.

5.4 At a Glance

- With the help of the metaprompting approach, optimized prompts can be created for your own LLM or for third-party systems.
- Metaprompting is an iterative process that step by step generates a new prompt or optimizes an existing prompt.
- In metaprompting, the system itself questions the quality of the prompt with regard to the task to be solved and independently offers ideas for improvement or refines the prompt by asking the user further clarifying questions.

- The Valisory Prompt Studio combines all the mechanisms of metaprompting presented here into a fully automated prompt that can both optimize existing prompts and create entirely new ones.
- Metaprompting is a powerful tool. However, it should only be used for comprehensive prompts, as it is oversized for short requests to the system.

6

Reasoning Models

Abstract Following the great success of LLMs, the so-called reasoning models are now also gaining ground. These are specially optimized LLMs that are capable of deeply engaging with a question within a given context before providing an appropriate answer. This chapter first provides a general overview of reasoning models and their functionality, then discusses the specifics of prompting, and introduces a prompt framework for reasoning models.

The use of LLMs for text generation has already become routine in many fields. Since the introduction of ChatGPT o1 and, at the latest, the release of DeepSeek-R1, the so-called reasoning models have been gaining increasing importance. These models are capable of deeply engaging with users' questions, first developing a highly structured approach to problem-solving before providing the actual answer. The following sections will first provide a detailed introduction to these reasoning models. Subsequently, the specific features of prompting for reasoning models will be discussed, and finally, a specialized prompt framework for reasoning models will be described.

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_6

6.1 The Background of Reasoning Models

In the course of research and further development of large language models, the so-called reasoning models have been developed as a specialization. Their specialization lies in their enhanced ability to solve complex tasks and logical problems. These models are generally very large; for example, some are based on more than 100 billion parameters.

As the name of this category of models suggests, their goal is to utilize a reasoning process to solve problems. Instead of "just" producing meaningful words and sentences based on statistical probabilities, they are intended to understand problems and solve them in a structured manner through logical reasoning. They were not fundamentally redesigned for this purpose; rather, they differ from classic language models primarily in that they automatically employ various prompting techniques. Depending on the model, there can also be significant differences in the training process. The fundamental technique for reasoning models is Chain of Thought (see Sect. 4.3), which is specifically extended by various advanced approaches.

> Reasoning models can, for suitable problem types, sometimes deliver remarkably good results. Approaches such as inductive out-of-context reasoning (Treutlein et al., 2024) demonstrate that LLMs can infer indirect relationships from training data and verbalize them logically and correctly. To what extent this reflects a true understanding of context remains unresolved.

If these reasoning models are so adept at logically solving problems in a manner similar to human reasoning, why have they not replaced traditional language models?

Like most current language models, reasoning models require a significant amount of resources, which affects both providers and users. The effectiveness of these models is also due to their size. And as size increases, so do the costs for both training and operation. As in other areas of the language model technology stack, the relationship here is exponential. As capabilities increase, the demands for resources such as

energy or storage space also rise dramatically. Additionally, due to their size, these models are comparatively slow. Depending on the task, it can easily take several minutes for a reasoning model to respond to a prompt.

> The higher operating costs associated with reasoning models are also evident in the fact that advanced models in this category are currently included by various providers (such as OpenAI) only in higher-priced usage packages and sometimes incur significant additional costs in use (for example, due to high costs per token).

A significant factor here is the approach to task decomposition and synthesis. The additional reasoning process also requires corresponding resources. It can be imagined as if the reasoning model conducts an analysis for a task, independently writes several prompts, and then answers each of them in turn.

This approach introduces another factor influencing performance. The reasoning process generates tokens, which are then evaluated by the model in the next step. These tokens, in turn, fill the context window and thus take up space that could be used for the actual result of the query. One way to address this is to technically increase the context window, which again ties up more resources when processing prompts. Another approach adopted by language model developers is to introduce a dedicated token type for reasoning models, the so-called reasoning tokens, which are generated and then deleted during the reasoning process. As a result, they only temporarily occupy the context window. Another approach is to execute the reasoning process in the so-called *latent space*. This technical area of an LLM is a background memory that users of the language model cannot access. This makes it possible to represent reasoning tokens at least as numerical or binary data. Since users do not see these intermediate results of the reasoning process anyway, there is no need to generate human-readable text and then process it in the next step. This allows for the elimination of a processing step. Moreover, this approach opens up the possibility of designing an entirely new storage model for reasoning tokens that operates even more efficiently.

6.2 Use Cases for Reasoning Models

Reasoning models are therefore currently not the tool of choice for smaller or simpler tasks. They are still too expensive, too slow, and too resource-intensive for such applications. However, they are well suited for tasks involving complex interrelationships.

For example, they are used in social research to gain new insights from empirically collected data. The language model is provided with relevant information such as the research design, raw data, and previously obtained findings, and is then asked for an assessment or further analysis. In an experiment conducted in our academic environment, OpenAI's current reasoning model "o1" was able to identify a completely new correlation cluster in the data of a quantitative study, which had previously gone unnoticed by the research team. The validity of this cluster was subsequently verified mathematically and was particularly surprising to the team, as a complex mathematical problem was solved using a language model. The reasoning process took about 20 minutes.

Another area where reasoning models can be effectively applied is the analysis and review of documents such as contracts or budget plans. The capabilities of these models make it possible to uncover complex interdependencies or inconsistencies and to develop meaningful additions or adjustments. While this is fundamentally possible with traditional LLMs as well, a comparison shows that reasoning models develop a much deeper understanding, resulting in significantly higher quality outcomes. Since these models— as outlined in the research example— can also comprehend mathematical and statistical relationships, reasoning can also be applied to documents such as budget plans and similar materials. In an experiment we conducted with a municipal budget plan of about 700 pages, the model was able to identify concrete inconsistencies and opportunities for improvement after only a brief analysis.

The comprehensive understanding capabilities of reasoning models also enable their use beyond classic textual tasks. For instance, it has been demonstrated that they possess the ability to plan and optimize production processes. The task here is to allocate production orders as efficiently as possible to the available resources while optimizing

various metrics, such as average process throughput time. Experiments have shown that such optimization by LLMs can be on par with that achieved by neural networks specifically trained for this purpose. (Abgaryan et al., 2024)

The use of reasoning models in software development also leads to significantly better results. When writing code, more complex requirements can be solved effectively. Experiments show that these models almost always produce extensible, testable, and error-free code. While this was often the case with traditional LLMs as well, a comparison reveals that reasoning models optimize for aspects such as extensibility and maintainability, resulting in noticeably higher code quality overall. These models can also be used in software quality assurance. Many software teams have so-called code review processes, in which the written code is reviewed by one or more developers. Parts of this review can be automated using tools that leverage reasoning models. For example, common programming errors can be detected and directly corrected by the model.

Finally, the comprehension and analytical capabilities of these models are also utilized in search engines. Providers such as Perplexity, which combine internet search with the capabilities of LLMs to deliver directly understandable answers based on search results instead of just links to websites, often already offer reasoning models. As explained, these models require significantly more time for research, but they draw on a much larger number of sources and ultimately provide more precise and comprehensive answers.

Important

The capabilities of reasoning models in software development already clearly indicate how this profession will change in the future. Language models will increasingly take over and automate parts of the work. Just as it has long been common practice to receive code suggestions and to generate parts of code automatically using templates, programming with an LLM will become a standard part of the software developer's role in the future. However, language models will not replace the profession itself any time soon, as developing an application involves much more than just writing code.

The fact that LLMs are still rarely used in software development is partly due to the confidentiality and compliance issues that would arise from using public models such as ChatGPT, Claude, or Gemini, and the fact that running local models is still comparatively uneconomical. Another limiting factor is the size of the context window. Programs quickly become so large that the LLM can no longer process them in their entirety, which means that important knowledge required to produce maintainable code according to project requirements would be missing. Approaches such as Retrieval Augmented Generation (RAG) have so far only yielded unsatisfactory results.

6.3 Reasoning Techniques of the Models

What distinguishes reasoning models from traditional language models is primarily their autonomous use of various prompting techniques. The first step is the chain-of-thought technique, which has subsequently been further developed through a range of advanced approaches. How prompting changes when working with reasoning models is demonstrated in Sect. 6.4.

The following section describes various reasoning techniques used in modern models. An interesting observation emerges: when examining these techniques, it becomes clear that they are fundamentally not particularly complex, but rather closely mirror the human approach to problem-solving, having simply been made explicit for the models. Nevertheless, research clearly shows that such techniques, which may seem self-evident to us, have an extremely positive impact on the quality of LLM outputs.

6.3.1 Automatic Chain of Thought

The chain-of-thought technique (Sect. 4.3) forms the foundation of reasoning models. Since the models apply this automatically, it is also referred to as *Automatic Chain of Thought*. The idea is the same as with the manual technique: a problem is broken down into logical sub-steps, which are then solved consecutively by the model. The final step is

either the ultimate solution to the problem or, alternatively, a dedicated synthesis of the partial solutions.

Depending on how the model implements chain of thought, the reasoning process may be visible to the user and can be traced. For example, when searching with Perplexity, the reasoning is output during the process, with the model explaining what needs to be solved, how it structures the task, and what is then researched on the internet. Reasoning with o1 from OpenAI behaves similarly; here too, the model's "thought process" is displayed during execution. In the Deepseek R1 model, reasoning is generated as part of the response within special <think></think> tags.

6.3.2 Self-Consistency Decoding

Self-consistency decoding can be used to further improve the results of a reasoning process via chain of thought. For a given question, several (e.g., 3 or 5) reasoning paths are generated. The model then reviews the results of each path and selects the most probable overall answer. In effect, this approach has the model consider a problem multiple times, often in parallel.

For example, with this technique, a model would develop several step-by-step solutions to a mathematical problem rather than just a single answer. These solutions are compared, and the most logical and consistent result is selected. This reduces errors in understanding or approach.

6.3.3 Tree of Thoughts

A significant extension of chain of thought is the tree-of-thoughts technique (ToT). Instead of following a single, linear path of reasoning as in CoT, the model generates a tree of solution paths. This approach is comparable to mind mapping and is particularly useful when complex problems can be solved in different ways, depending on how the reasoning process unfolds and which assumptions are made along the way.

Experiments have shown that, for example, significantly better results can be achieved for Sudoku puzzles using tree of thoughts than with zero-shot, one-shot, or few-shot techniques.

6.3.4 ReAct

A technique building on this, which is primarily used in the context of autonomous AI agents, is the separation of reasoning (thought process) and action, known as ReAct. The underlying approach is essentially simple. The model is instructed first to use reasoning to understand the problem and plan a solution path. Only in the second step is the actual problem solved, which for AI agents may also involve interacting with other (AI) systems. Since this can potentially have irreversible effects on the environment (consider a scenario where an AI agent deletes something from a system), it makes sense to first structure and fully understand a problem rather than proceeding step by step without foresight.

6.4 Prompting for Reasoning Models

Unsurprisingly, the superior capabilities of reasoning models are also evident in their practical use. In general, the overarching principles of prompting language models still apply. Prompting is, by and large, very similar to that of classical LLMs. However, there are several specific limitations that must be taken into account.

Fundamentally, these models are adept at comprehensively analyzing problems. Therefore, it is often unnecessary to include a large amount of detail or numerous examples in the prompt. This does not mean that examples are not useful, but rather that they can be more superficial. The necessity and extent should always be assessed on a case-by-case basis. Such contextual information, as well as guidance on the desired solution approach, are especially important when there is a risk that the model might draw incorrect conclusions during the reasoning process. For instance, if a particular assumption can be ruled out in advance, explicitly stating this can help improve both the approach and the quality of the results.

Since reasoning models already employ chain-of-thought and related techniques for problem solving, it is not necessary to explicitly request these in the prompt. Prompting techniques aimed at eliciting such processes are therefore redundant. Similarly, specifying a step-by-step solution path is not advisable, as the models are trained to develop and refine these independently throughout the process. Forcing a particular approach can, in fact, measurably degrade the results.

Interestingly, the use of roles is also often not beneficial. Various experiments have shown that, in reasoning tasks involving chain-of-thought, imposing roles can actually lead to poorer outcomes—especially when the assigned role does not align with the one required for effective problem solving. The recommendation, therefore, is to avoid using roles with reasoning models (Han & Wang, 2024).

Prompt Framework for Reasoning Models

As previously described, prompting reasoning models follows somewhat different rules. Nevertheless, a prompting framework has emerged in this context that leads to very good results. This framework consists of four sections:

1. **Goal:** This section provides a detailed description of what the prompt is intended to achieve.
2. **Return Format:** Here, the structure in which the LLM should present its output is specified.
3. **Warnings:** This section outlines potential sources of error to be avoided and provides restrictive instructions.
4. **Context Dump:** The final section provides background information necessary for an optimal solution to the task. This helps to ensure that the response is placed in the correct context.

Prompts structured in this way and enriched with relevant information and instructions promise optimal results, as they provide the model with a clear but not overly restrictive framework while leaving sufficient room for the model's own "thought process."

Below is an example prompt based on the framework presented here.

Example

1. **Goal**
   I require a detailed review of current best practices and trends in the field of hybrid work. The focus should be on proven strategies for organizations that combine both remote and in-office work. The objective is to provide a well-founded basis for decision-making regarding the further development of our internal work policies.
2. **Return Format**
   Create a structured summary with the following sections:
   - **Definition**—A clear explanation of what hybrid work is.
   - **Advantages and Challenges**—A list of the key benefits and challenges for organizations and employees.
   - **Current Trends**—An overview of recent developments and innovative approaches.
   - **Best Practices**—Concrete strategies that successful organizations use to implement hybrid work effectively.
   - **Sources and Further Reading**—A list of reputable studies, reports, or articles for further exploration.
3. **Warnings**
   - Ensure that all information is drawn from **current and reputable sources** (e.g., academic articles, studies, reports from consulting firms or tech companies).
   - Avoid general or outdated statements—the review should be **state-of-the-art**.
   - The summary should be **concise and clear** (max. 1000 words) so that it can serve as a decision-making aid for executives.
4. **Context Dump**
   Our company is planning to revise its existing remote work policies and develop a long-term strategy for hybrid work. In doing so, both employee needs and operational requirements should be taken into account. Particular emphasis is placed on aspects such as productivity, corporate culture, digital tools, and long-term workplace design.

References

Abgaryan, H., Harutyunyan, A., & Cazenave, T. (2024). Llms can schedule. *arXiv preprint arXiv:2408.06993*. https://arxiv.org/pdf/2408.06993v1. Accessed: 31. Jan. 2025.

Han, Z., & Wang, Z. (2024). Rethinking the role-play prompting in mathematical reasoning tasks. In *Proceedings of the 1st Workshop on Efficiency, Security, and Generalization of Multimedia Foundation Models* (Pages 13–17). https://dl.acm.org/doi/pdf/10.1145/3688864.3689149. Accessed: 31. Jan. 2025.

Treutlein, J., Choi, D., Betley, J., Marks, S., Anil, C., Grosse, R. B., & Evans, O. (2024). Connecting the dots: Llms can infer and verbalize latent structure from disparate training data. *Advances in Neural Information Processing Systems, 37,* 140667–140730. https://arxiv.org/pdf/2406.14546. Accessed: 31. Jan. 2025.

7

AI Agents

Abstract This chapter begins by introducing the concept of AI agents. It is important for companies to engage with these systems, as they will be able to autonomously perform many internal and external tasks and processes in the future. Various AI agent concepts are presented, along with descriptions of how agents interact with each other and with other systems, exchanging data and information. The chapter then explains why prompt engineering is so crucial in this context. Finally, different use cases for AI agents in business are presented.

Many current applications of language models focus on dialogic and interactive scenarios. A user proactively interacts with a system such as ChatGPT or Claude via a chat interface. The interaction takes place at the moment when the user wants to solve a specific problem. Working with the LLM in this way is therefore synchronous.

A development that is currently gaining relevance for companies is that of AI agents: (semi-)autonomous language models that independently and without ongoing user interaction perform tasks and produce results in the background. The concept and components of agents are presented in this chapter in the context of prompt engineering.

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_7

7.1 The Concept of AI Agents

Language models such as ChatGPT, Claude, or Gemini are currently used primarily in a reactive manner. Employees can use them to fully or partially automate tasks and quickly achieve good results. Companies can thus increase their efficiency and, in some cases, also improve quality. The associated reduction in the time required for activities ultimately leads to cost savings. This effect arises mainly indirectly through the combination of many small optimizations along the company's value chain.

Although it is extremely worthwhile for companies to invest in these targeted efficiency improvements enabled by language models, this type of AI holds even greater potential for businesses. Imagine being able to leverage the capabilities of LLMs not just at isolated points in the value chain, but to hand over entire business processes to language models that, within a defined framework, independently make intelligent, well-founded decisions and also autonomously carry out the necessary activities. This would create an automation of business processes that would not initially require any technical expertise or custom software development. Instead, it would be based purely on domain knowledge, where the initial situation, the goal, and, if necessary, parts of the approach are explained to the LLM in natural language. There are clear parallels here to citizen development, which also aims to empower people to create automations and software without fundamental programming knowledge. This is often achieved with low-code or no-code systems, where standard use cases (e.g., querying data from a database or preparing information in tables and charts) can be assembled with a few clicks. The major difference from citizen development is that with AI agents, no technical knowledge is required at all. In addition, the automation is executed autonomously and not deterministically.

At first glance, one might therefore assume that AI agents are superior to low-code or no-code systems. In reality, however, it depends on the specific situation, as both types of systems have their challenges and limitations. Only by combining the two approaches does an invaluable potential emerge for companies. Just as some processes are very

strict and deterministic—making them easy and efficient to implement in software—others require more intelligent, discretionary decisions, which is a clear use case for AI agents.

7.1.1 Definition of AI Agents

Unsurprisingly, there is no universally accepted definition of AI agents. In 1995, the term was first defined in the context of AI, describing an agent as something that perceives its hrough actuators (Russel & Norvig, 1995).

This definition is rather abstract, not very tangible, and does not include the development of LLMs, which is the focus of this book. In the context of such language models, we therefore define AI agents as follows:

At its core, an AI agent is a language model (LLM) that has been equipped with the ability to search for relevant information, store it (memory), and interact with the outside world (other systems or AI agents).

An LLM equipped with such capabilities is referred to as an augmented LLM. The first key capability of such an augmented LLM is the ability to independently search for information. The model formulates appropriate search queries and searches available sources for relevant results. For publicly available models such as Claude by Anthropic, this means using an internet search engine. The augmented LLM can then process the search results and incorporate them into the task at hand.

The second capability is the selective storage of knowledge as memory. This knowledge is not lost when the context window changes (see Sect. 1.3), but remains available for future actions of the language model. Selective storage means that memory is not a simple repository where all intermediate results are stored. Rather, the model independently decides which information is necessary for the future and discards the rest.

The third component enables AI agents to interact at their own discretion with other systems and tools. These can, for example, be other AI agents to which tasks are delegated. The AI agent is aware of

the available systems, their capabilities and limitations, and can independently decide whether and when to use them.

> Strictly speaking, the ability to conduct research is also an integration with a third-party systebm and could therefore be subsumed under the last component. However, since this research capability often represents a fundamental aspect of tasks for AI agents and working with third-party systems can also have side effects, the search capability is considered a separate component.

The fact that an LLM is at the core of the AI agent is an important aspect for its learning ability. AI agents learn during their operation and, over time, produce different—hopefully better—results. In some cases, agents are also reactive; they respond to changes in their environment and adjust their behavior accordingly. On the other hand, they are also proactive, not only pursuing their goals independently but also planning their actions in advance.

7.1.2 Distinction Between Agents and Workflows

From an architectural perspective, agent systems are fundamentally distinguished between workflows and agents. The distinction is based on the use of software code.

An AI agent is an autonomous system in the sense that it independently determines the steps required to solve a task, executes them, and produces the results. It selects the necessary tools at its own discretion and meaningfully integrates the individual results.

A workflow, on the other hand, is a system that consists of a combination of one or more language models and software tools, orchestrated by code. For orchestrating individual components, a no-code or low-code system can be used, for example. Thus, a workflow system always contains a certain degree of determinism.

The definitions of workflow and AI agent are not mutually exclusive. AI agents can be used within a workflow system and be controlled by

the system, just as an AI agent can utilize a workflow system to accomplish tasks.

7.2 Architectures of Workflow Systems

There are various approaches to the architecture of workflow systems, all of which are fundamentally based on established architectural patterns from software development. Depending on the objective, different components (LLMs and software modules) are arranged in a sequence and then executed either sequentially or in parallel. The decision regarding the exact sequence of a workflow can be made either by software code or by LLMs themselves.

7.2.1 Workflow Architecture 1: Iterative Quality Control

In this architecture, two LLMs are combined to process a task and ensure the quality of the result. The first LLM handles the actual task using its capabilities. If this LLM is a KI agent as defined, this could mean, for example, that it conducts an internet search for information or interacts with systems within the company.

Once the results have been produced, the first LLM passes them on to the second LLM. This second language model specializes in ensuring the quality of the results. It analyzes the output and compares it with the original task. If the desired goal has not yet been achieved, it generates feedback for the first LLM and passes this feedback back to it.

The first LLM then resumes processing the task, incorporating the feedback. This cycle continues until the results are satisfactory and the workflow is completed with the achievement of the objective.

For an effective interaction chain, the second LLM is used solely as an additional module for quality assurance and is only utilized by the primary LLM for this purpose. The final results are then passed on by the first LLM to the user or to the next workflow stage. In software architecture, this pattern is also known as a “sidecar.”

7.2.2 Workflow Architecture 2: Selective LLM Selection

Different language models—whether large language models or small language models—have varying strengths and weaknesses. Since these models are not deterministic, it can be beneficial in complex systems not to predefine which language model to use, but rather to make this decision dynamically within the workflow.

This architectural pattern incorporates a specialized language model as a router. The model's task is to receive information and decide which subsequent language model can best address the task with or for this information. The router LLM then forwards the task to the appropriate model, which produces the final result.

For example, a prompt may require an internet search. The router knows which LLM has this capability and can select among them. A second prompt might be a translation task, which does not require an internet search but does require a model specialized in translation. The router LLM would then forward the task to such a model.

7.2.3 Workflow Architecture 3: Parallelization with Multiple LLMs

For time-consuming or complex tasks, multiple LLMs can also be used, but unlike selective selection, the task is processed in parallel. In contrast to the selective selection architecture, there is no router LLM; instead, a programmed component distributes the task to several LLMs simultaneously.

The individual results are then consolidated into a final result by an integrator. In this architectural pattern, the integrator is not an LLM but a software component, and is therefore deterministic.

Parallelization can occur in two ways. If the overall task is sufficiently complex, it can be broken down into sub-tasks. These sub-tasks are then assigned to different LLMs and processed simultaneously. In this case, the LLMs may lack the overall context, especially if they are specialized language models that cannot grasp the entire task.

In the second approach, the entire task is distributed across multiple LLMs, and the integrator selectively combines the results. In the simplest case, it checks which partial result occurs most frequently, selects this as the final result, and discards the others.

7.2.4 Workflow Architecture 4: Multi-LLM Prompt Chaining

Within the prompt framework of prompt chaining (see Sect. 3.1), it has already been suggested that this technique allows requests to be distributed across multiple language models, which are then processed sequentially, with each model further developing the result. In a workflow system, this chaining of language models occurs automatically, and manual interaction with multiple models by the user is not required.

Two additional modules can be integrated into the process for control purposes. First, it may be necessary to deterministically modify intermediate results. For this, code-based transformers are used, which are positioned between two LLMs and modify the output of one before passing it to the next. Second, there may be situations where a workflow should be terminated prematurely. For this, conditional modules can be inserted, which are also programmed with code. These modules decide, based on the result of an LLM, whether the workflow should continue. If so, they forward the result accordingly; if not, they terminate the workflow.

7.2.5 Workflow Architecture 5: Orchestration with LLMs

As a variant of parallelization, where a programmed component distributes the task to multiple LLMs in parallel and a programmed integrator handles the consolidation, LLM orchestration delegates the entire responsibility to two LLMs.

The first LLM, the orchestrator, is responsible for dividing and parallelizing the task. It determines how and with which LLMs the task is parallelized and forwards the respective (sub-)tasks to the models. The

second LLM, the synthesizer, receives the results from the various LLMs and combines them wholly or partially, depending on the requirements and its own discretion.

In orchestration, the sequence and combination of LLMs are entirely managed by a controlling language model, just as the final result is produced by a model. Depending on the implementation, this can be either the orchestrator or the synthesizer.

7.2.6 Combining Architectures

These different architectures of LLM workflows should be understood as building blocks and ideal-typical patterns. Workflow systems used in practice are often not precisely attributable to a single architectural pattern, as they include detailed adaptations. For example, programmed filter systems or monitoring components may be installed that are not part of the original pattern.

On the other hand, architectural patterns can be combined. For instance, within a workflow parallelized by an orchestrator, chained LLMs can be used, with occasional quality-assuring sidecar LLMs integrated as needed.

As with the architecture of software systems, the goal is to use proven and documented approaches on the one hand, but above all to achieve the business objective on the other.

7.3 Architectures of KI Agents

Unlike workflow systems, KI agents are limited in their architecture to a single augmented LLM with the corresponding capabilities. However, they can also be part of a workflow system.

As mentioned in the definition of KI agents, they are fundamentally capable of interacting with other systems and KI agents. However, this collaboration does not constitute a workflow system, as the latter are characterized by a black-box nature to the outside. Furthermore, a KI agent is autonomous and non-deterministic, and can therefore

independently decide whether and with which next component or KI agent to proceed. Multi-KI-agent systems are, in essence, a collection of autonomous KI agents that collaborate as needed, but ultimate responsibility for completing the task always lies with a single agent.

The work of KI agents always begins either directly or indirectly through interaction with a user. Directly, for example, when someone submits a request to a chatbot. Indirectly, for instance, when a software component is programmed to start the agent at specific times (e.g., daily).

While processing a task, agents can also interact with users, for example, by pausing to request feedback or to have questions answered. In practice, this could occur via a persistent chatbot or through email communication. Additionally, software components can provide information or send signals to a working agent, influencing task processing.

Once the KI agent's task is fully completed, the agent is terminated. However, the task may not have a clearly defined end, for example, if the agent is to perform the same activity regularly for the foreseeable future.

7.4 The Necessity of Prompt Engineering for AI Agents

The autonomy and extensive capabilities of AI agents make it essential to prepare and configure them carefully. Since they are "programmed" using natural language, structured, effective, and goal-oriented communication in the form of task instructions is crucial. In short, effective prompt engineering is indispensable for AI agents.

Fundamentally, the prompt itself must first be written for the agent. The prompt frameworks presented in this book can help to define the agent's task and objective precisely, thereby avoiding undesired outcomes or side effects.

In addition, the agent must be informed about which resources, such as other agents and tools, are available. There are both technical and natural language approaches to this. Capabilities such as memory

storage and internet search are typically taught to the models technically and do not require any additional configuration.

Furthermore, the context of the agent is extremely important. It must be determined precisely which information is needed and in what format, so that the agent can effectively fulfill its task. This goes beyond simply writing the prompt, encompassing structure and format. Since no additional information or data can be provided to an active agent after deployment, these must be included as part of the prompt or supplied as documents during configuration. Unlike interactive chats with an LLM, careful pre-planning is significantly more important for agents.

An aspect that does not arise in interactive work with language models is testing. Configured agents must be thoroughly tested before deployment to ensure that, on the one hand, they achieve the intended goal, and on the other, do not produce unwanted side effects in their output. A good approach to verifying functionality is A/B testing: several different versions of the prompt and context are written or produced to determine which works most effectively. In the simplest case, testing means running the LLM in a classic, non-agent mode with the prompt and validating the results. Additionally, different LLMs can be used. A high-quality prompt can be created using metaprompting. This prompt can then be reviewed by a second LLM for criteria such as clarity, completeness, and coherence. The results produced by the agent LLM during testing can in turn be evaluated by another language model. From this interplay, combined with the skills and knowledge of the prompt engineer, effective prompts for the agent can be developed that deliver the desired outcome at the expected level of quality.

7.5 Typical Use Cases for AI Agents in Companies

The abstract representation of the possible combinations of workflows and agents, especially in comparison to no-code and low-code systems, demonstrates that the range of use cases can be extensive. In principle,

for fully or partially digitized business processes, it is possible to assess which opportunities exist for an agent-based approach. Moreover, agents open up many new possibilities for digitalization and automation due to their flexibility and ability to understand natural language—capabilities that were not possible with previous tools.

7.5.1 Automated Customer Support

For some time now, companies have been seeking to automate routine tasks in customer support. The aim is to relieve existing staff and make better use of available resources. A commonly used tool for this purpose is chatbots, which are, for example, embedded on a website and have access to help articles and FAQs (Kohne et al., 2020). Users can ask these chatbots questions and receive answers based on the available information.

These approaches have faced two main challenges. On the one hand, the company must create and provide the necessary knowledge. On the other hand, chatbots were for a long time only of limited use, as they did not rely on language models but instead searched their existing knowledge base for keywords from user queries. This often led to incorrect or insufficient results. Furthermore, no real interaction was possible, as the chatbot was essentially just an improved search interface for information that was already available.

This changed abruptly with the introduction of LLMs. When properly trained and configured, chatbots can now understand user queries and provide accurate answers in the correct context. They can also respond to follow-up questions and provide relevant source references. In addition, the chatbot is not limited by business hours and is therefore available at all times. Multilingual support can also be integrated with minimal effort.

Taking this a step further and transforming the "simple" chatbot into an AI agent offers users significantly greater value. The agent can analyze and process a much broader range of company information using its search capabilities. Combined with appropriate authorization and authentication, the agent could not only provide answers but also

process requests directly, such as opening customer tickets, managing contracts, or updating master data.

7.5.2 Automated Recruiting

Recruiting often involves repetitive tasks that are also fundamentally suitable for the use of AI agents. For example, the screening of incoming applications can be automated. An agent can objectively evaluate candidates without emotional bias and identify the best candidates for an open position. Applications that do not meet the requirements can be automatically rejected or reviewed by HR staff before rejection. For positive decisions, agents can interactively schedule interviews with candidates, taking into account the availability of relevant employees. After hiring, AI agents can support onboarding, explain company processes and policies, answer questions, and, for example, identify and automatically involve other employees with relevant expertise.

Due to the significant potential in recruiting, many of these capabilities have already been implemented in platforms and tools for some time. With the capabilities that AI offers in combination with agent systems, these processes can be made even more user-friendly, faster, and more autonomous—resulting in cost and efficiency gains for the company.

> Because LLMs depend on their training data and learn over time, it must be clearly stated that a language model is no more capable of making an objective decision than a human would be in the process.

7.5.3 Intelligent Document Management

AI agents can also provide valuable support and automation in handling documents within companies. Documents received through various channels (email, scanners, chat messages, document repositories, etc.) can be centrally processed and stored. For example, the agent can check documents for compliance and data protection guidelines and

classify them accordingly. Assessments made by the LLM can also be stored alongside the document, enabling, for instance, an evaluation of whether the document might be part of a fraud attempt.

Content can be extracted both verbatim and in terms of meaning and stored for use in other systems. Building on this, summaries can be generated and made available to relevant employees. As in a classic document management system (DMS), this makes content searchable. What distinguishes the agent from a DMS is its greater interactivity and understanding of content. For example, within a DMS, an agent could regularly generate a change report with an assessment of modifications based on version control. If, for instance, contract details change, the agent can highlight these changes and interpret them in comparison to the previous version.

Furthermore, AI agents can assist with archiving by identifying documents that are no longer needed and flagging them for archiving or archiving them automatically. New versions of documents can be found in the DMS, but are often stored decentrally within the company and can then be automatically consolidated and updated to the latest version in a central location.

7.5.4 Dynamic IT Security

The security of IT infrastructure is an issue that is becoming increasingly important as processes become more digitalized. In the field of IT security, AI has already been used in various forms for some time, for example, to detect behavioral anomalies or to classify risk levels for software and hardware.

Building on such systems, an AI agent can autonomously handle security incidents by assessing their relevance and initiating appropriate countermeasures. IT systems can be automatically isolated from the rest of the infrastructure and analyzed to examine the behavior of potential attackers or malware.

Based on this, the security of the infrastructure can then be automatically adjusted, for example, by applying updates and patches or shutting down systems. The company's defense strategy is adapted and

continuously improved according to the situation. As a result, IT security becomes more adaptive and responsive, and can potentially respond more effectively to new threats.

Many of these approaches have already been implemented on a small scale and in individual components. Here, too, an AI agent can combine and control existing approaches and systems with its capabilities—but it would not replace an intrusion detection system (IDS). Moreover, language models and thus AI agents have limitations and challenges, so blind trust in this technology—especially in such a critical area as IT security—is currently not advisable.

7.6 At a Glance

- AI agents are (semi-)autonomous systems that can use LLMs to perform tasks fully automatically and without human interaction.
- By leveraging LLMs, entirely new possibilities for automating tasks and processes become available.
- AI agents can be created and controlled using prompts. In addition, they often require integration capabilities and interfaces to third-party systems or other LLMs.
- A workflow system for automating tasks and processes follows a predefined, deterministic path and completes the individual steps until the desired result is achieved or an error occurs. In contrast, AI-based agents autonomously develop a solution path for a given task and then execute it automatically.
- For an AI agent to work effectively, a very precise prompt is required for control. The tips and frameworks from the field of prompt engineering presented earlier can be helpful here.
- AI agents can be used for a variety of purposes within a company. These range from automated customer support using intelligent chatbots, to support in recruiting, to automated document management and assistance in IT security.

8

Outlook

Abstract This chapter concludes with an outlook on the future of AI systems. It becomes evident that AI functionalities will increasingly be integrated into a wide range of systems, and that AI will soon directly or indirectly impact all areas of life. It is therefore all the more important to engage with the technology and its value-adding applications today.

The current pace of development and adoption of AI systems is likely to continue in this form for another two to three years. During this period, advances in training methods and the availability of ever-larger datasets will enable these systems to make significant leaps forward. According to various sources, AI-based systems will influence or even control all areas of life worldwide by 2030 at the latest (e.g., Federal Ministry of Labour and Social Affairs, 2023; Opiela et al., 2018; Straits Research, 2024).

Even today, AI technologies are integrated into far more systems than many people realize. In the field of research as well, major breakthroughs are expected in the coming years that would not be possible without AI.

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_8

AI will increasingly be able to take over tasks within companies. Starting with simple, repetitive tasks, the rise of AI agents—which is already clearly emerging—will soon make it possible to fully automate complex processes and workflows both within organizations and at external interfaces with customers, partners, and suppliers, and to do so with high quality. In times of skilled labor shortages, this is certainly one of the greatest opportunities.

Despite all the positive opportunities and possibilities, it is important to remain aware that, like any technology, AI is a double-edged sword. Issues such as data protection, misuse, and intellectual property rights will undoubtedly occupy the courts for years to come.

It is crucial to engage with this technology now in order not to fall behind. One key competency on the path to successful AI adoption is prompt engineering. Even though future AI systems will undoubtedly become better at understanding user intent, a fundamental understanding of the technical principles and functioning of AI systems, as well as the ability to formulate effective queries—as described in the introductory light switch metaphor—will remain essential for navigating an increasingly fast-paced and technologically advanced world.

With all the possibilities that AI already offers us—and will certainly continue to offer in the future—it is important never to stop critically questioning the results and thinking independently. For all their technical sophistication, AI systems (at least for the foreseeable future) cannot demonstrate true creativity. This remains—for now—the exclusive domain of humans.

At this point, we conclude this specialist book with the famous quote from Immanuel Kant, who, during the Age of Enlightenment in 1784, coined the following phrase: "Sapere aude!"—"Have the courage to use your own understanding!"

We wish you every success on your personal journey with AI. Should you have any questions or require support in the field of AI, please feel free to contact us at any time.

References

Bundesministerium für Arbeit und Soziales – Denkfabrik. (2023). Die Arbeitswelt der Zukunft: 5 Szenarien zur „Mensch-Technik-Interaktion 2030". https://www.denkfabrik-bmas.de/rubriken/wissen/die-arbeitswelt-der-zukunft-5-szenarien-zur-mensch-technik-interaktion-2030. Zugegriffen: 31. Jan. 2025.

Opiela, N., Kar, R. M., Thapa, B., & Weber, M. (2018). Exekutive KI 2030 – Vier Zukunftsszenarien für künstliche Intelligenz in der öffentlichen Verwaltung. Kompetenzzentrum Öffentliche IT. https://www.oeffentliche-it.de/documents/10181/14412/Exekutive+KI+2030+-+Vier+Zukunftsszenarien+f%C3%BCr+K%C3%BCnstliche+Intelligenz+in+der+%C3%B6ffentlichen+Verwaltung. Zugegriffen: 31. Jan. 2025.

Straits Research. (2024). Künstliche Intelligenz (KI) Marktgröße, Anteil und Trends Global Report 2030. https://straitsresearch.com/de/report/artificial-intelligence-market. Zugegriffen: 31. Jan. 2025.

9

Copy Templates

Abstract This chapter provides templates corresponding to the models and frameworks presented. They can serve as daily companions for prompting, for example, by being placed directly next to the keyboard.

9.1 The Valisory Prompt Quality Model

The Valisory Prompt Quality Model		
Criterion	**Relevance**	
Factors	Question Formulation	Is the prompt clearly formulated and does it specifically request the desired information or action?
	Specificity	Are the requirements and objectives of the prompt specific enough to generate relevant responses?
	Comprehensibility	Is the prompt simple and clear enough for the model to correctly interpret the request?

(Fortsetzung)

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2_9

The Valisory Prompt Quality Model		
Criterion Factors	**Creativity and Originality**	
	Innovation Potential	Does the prompt encourage the model to go beyond general knowledge and find creative or innovative solutions?
	Inspiration	Does the prompt inspire original ideas or perspectives that are not trivial or obvious?
	Variability	Does the prompt provide sufficient scope for different answers or solution paths?
Criterion Factors	**Linguistic Quality and Style**	
	Clarity	Is the prompt clear and precise, without ambiguous or misleading wording?
	Style Adaptation	Is the style of the prompt (formal, informal, technical, etc.) appropriate for the objective and audience?
	Structure	Is the prompt logically structured and does it encourage a structured response?
Criterion Factors	**Target Group Appropriateness**	
	Understanding the Target Group	Does the prompt take into account the prior knowledge, interests, and needs of the target group?
	Addressing the Audience	Is the tone and directness of the prompt appropriate for the target group?
	Usefulness	Is the type of response sought by the prompt practical and of interest to the target group?

9.2 Prompt Frameworks

Prompt Frameworks			
No.	Name	Meaning	Process
1	RTF	Role-Task-Format	1. Description of the role 2. Description of the task 3. Description of the formatting
2	TAG	Task-Action-Goal	1. Description of the task 2. Description of the action 3. Description of the goal

(Fortsetzung)

Prompt Frameworks			
No.	Name	Meaning	Process
3	BAB	Before-After-Bridge	1. Description of the initial situation 2. Description of the target situation 3. Development of a bridge between the current and target state
4	CARE	Context-Actions-Result-Example	1. Description of the context or initial situation 2. Description of the actions to be taken 3. Description of the result 4. Description of an example
5	RISE	Role-Input-Steps-Expectations	1. Description of the role 2. Description of the data provided 3. Description of the necessary steps 4. Description of the expected outcome

9.3 The Expert Prompt Framework

Prompt

Act as a [expert role].

I require [description of the result].

You will [detailed task description].

In doing so, you will [further details].

Please [list exceptions].

Develop the result for [target audience].

Deliver the final result as [format].

Here is an example: [example].

9.4 The Valisory Prompt Studio

Prompt

You are the Valisory Prompt Studio, an experienced prompt engineer specializing in developing effective prompts for language models. You assist the user in creating the best possible prompt for their needs.

The user can either develop a completely new prompt or improve an existing one.

The following describes the procedure, which you must not deviate from.

The process is fundamentally iterative: through questions and feedback, the prompt is refined step by step.

At no point do you execute or respond to the developed prompt yourself.

You communicate in a formal and polite manner. The language of interaction with the user is the respective national language (unless otherwise specified). The language of the prompt to be created is also the national language, unless the user explicitly requests otherwise or provides a prompt for improvement that is written in another language.

The following description is divided into sections (A, B, and C). Start with section A.

A: Your general procedure

- If the user has not specified it, first ask which language model the prompt is to be developed for.
- If the user has not specified it, secondly ask whether a new prompt is to be developed (continue with B) or an existing prompt is to be improved (continue with C). Then proceed with the respective section below.

B: If a prompt is to be improved:

1. Ask for the prompt to be improved, if the user has not already provided it.
2. Analyze the prompt and identify its objective. Ask the user whether the objective you have determined is correct or if the user would like to define a different objective.
3. Continue with step 3 of section C.

C: If a prompt is to be developed:

1. Ask for the objective of the prompt. Ask follow-up questions if the user's answers are unclear or not precise.
2. Develop a first version of the prompt. Do not output this version, but proceed directly to step 3.
3. Analyze the current prompt from two perspectives:
   a. As a critical prompt reviewer
   b. As a creative prompt developer
4. Always provide the following:
   – The objective of the prompt
   – The current prompt
   – Suggestions for improvement from the perspective of the critical prompt reviewer
   – Relevant questions (maximum 3) for improving the prompt from the perspective of the prompt developer
5. The user will answer the questions. Ask follow-up questions if the answers are unclear, imprecise, or not meaningful.
6. Refine the prompt using the answers to the questions and any additional comments from the user.

References

Bundesministerium für Arbeit und Soziales Denkfabrik. (2023). Die Arbeitswelt der Zukunft: 5 Szenarien zur Mensch-Technik-Interaktion 2030. https://www.denkfabrik-bmas.de/rubriken/wissen/die-arbeitswelt-der-zukunft-5-szenarien-zur-menschtechnik-interaktion-2030. Accessed: 31. Jan. 2025.

Kohne, A., Kleinmanns, P., Rolf, C., & Beck, M. (2020). Chatbots. *Aufbau und Anwendungsmöglichkeiten von autonomen Sprachassistenten*. Springer Vieweg.

Opiela, N., Kar, R. M., Thapa, B., & Weber, M. (2018). Exekutive KI 2030 Vier Zukunftsszenarien für künstliche Intelligenz in der Öffentlichen Verwaltung. Kompetenzzentrum Öffentliche IT. https://www.oeffentlicheit.de/documents/10181/14412/Exekutive+KI+2030+-+Vier+Zukunftsszenarien+f%C3%BCr+K%C3%BCnstliche+Intelligenz+in+der+%C3%B6ffentlichen+Verwaltung. Accessed: 31. Jan. 2025.

Ramlochan, S. (2024). System prompts in large language models. Prompt Engineering & AI Institute. https://promptengineering.org/system-prompts-in-large-language-models/. Accessed: 31. Jan. 2025.

Russel, S. J., & Norvig, P. (1995). *Artificial Intelligence: A modern approach*. Prentice Hall.

Straits Research. (2024). Küstliche Intelligenz (KI) Marktgröße, Anteil und Trends Global Report 2030. https://straitsresearch.com/de/report/artificial-intelligence-market. Accessed: 31. Jan. 2025.

D. Koch et al., *Prompt Engineering in the Enterprise—An Introduction*,
https://doi.org/10.1007/978-3-658-49218-2

Turing, A. M. (1950). Computing machinery and intelligence. Mind, New Series, *59*(236) (Okt., 1950), Pages 433–460.

Vaswani, A. (2017). *Attention is all you need*. Advances in Neural Information Processing Systems.

Sources

Abgaryan, H., Harutyunyan, A., & Cazenave, T. (2024). Llms can schedule. *arXiv preprint arXiv:2408.06993*. https://arxiv.org/pdf/2408.06993v1. Accessed: 31. Jan. 2025.

Han, Z., & Wang, Z. (2024). Rethinking the role-play prompting in mathematical reasoning tasks. In *Proceedings of the 1st Workshop on Efficiency, Security, and Generalization of Multimedia Foundation Models* (Pages 13–17). https://dl.acm.org/doi/pdf/10.1145/3688864.3689149. Accessed: 31. Jan. 2025.

Treutlein, J., Choi, D., Betley, J., Marks, S., Anil, C., Grosse, R. B., & Evans, O. (2024). Connecting the dots: Llms can infer and verbalize latent structure from disparate training data. *Advances in Neural Information Processing Systems, 37,* 140667–140730. https://arxiv.org/pdf/2406.14546. Accessed: 31. Jan. 2025.