Tampereen yliopisto

Henri Heimonen

LARGE LANGUAGE MODELS IN BUSINESS ANALYTICS

A Case Study

Master’s Thesis
Faculty of Management and Business
Examiner:
Pekka Abrahamsson
January 2024

ABSTRACT

Henri Heimonen: Large Language Models in Business Analytics: A Case Study
Master's Thesis
Tampere University
Master's Degree Education in Management and Information Technology
January 2024

Artificial Intelligence (AI) became one of the most discussed topics in professional context soon after the launch of ChatGPT. Suddenly there was an approachable way to communicate with the AI. In business new disruptive moments tend to raise a question how this new thing could be embraced to advance business targets.

This study clarifies the possibilities of AI for business analytics. The goal is to understand if it is possible to extract summaries and detect sentiments from given, large texts. During the study, synthetic data is being used. Another target is to create a Proof-of-Concept (POC) software that demonstrates above-mentioned capabilities.

Reference software by Microsoft provided fast way forward, and enabled concentrating onto actual research questions, instead of building mere boilerplate. So called RAG-pattern (Retrieval Augmented Generation) was used. That enabled combining own data with capabilities provided by pre-trained AI model. Approach close to Design Science Research (DSR) was used to build the software iteratively and study the possibilities. In addition, narrative strategy was utilized to understand the DSR methodology itself deeper.

According to study AI can extract summaries and detect sentiments reasonably well. Recursion and handling larger texts in parts was used as a way around the limited size of the context window. Parallelism was utilized to increase efficiency. Few separate concepts were introduced to facilitate the handling of parts and maintaining the context awareness. One of these is a new idea coined as "base question". For DSR process few observations were made, in addition it was proposed that aspect of agile development could be brought into process.

As future topics, one suggestion is advancing the idea of base question further and maturing it. It is also proposed to understand how to avoid the overemphasis of beginning of the larger text in summaries, and whether companies integrating AI into office tools could provide a way to use modified "system prompt" – behavioral guidance – for certain use cases. As a final managerial implication, the recommendation to use commercially available tools is made because of rapid development.

Keywords: Artificial Intelligence, Retrieval Augmented Generation, Design Science Research, Text summarizing, Sentiment analysis, Base question

TIIVISTELMÄ

Henri Heimonen: Suuret kielimallit yritysanalytiikassa: tapaustutkimus
Diplomityö
Tampereen yliopisto
Johtamisen ja tietotekniikan maisterikoulutus
Tammikuu 2024

Tekoäly nousi nopeasti työelämän suosituimpien puheenaiheiden joukkoon pian ChatGPT:n julkaisun jälkeen. Yhtäkkiä saatavilla oli lähestyttävä tapa kommunikoida tekoälyn kanssa. Yritysmaailmassa tämänkaltaiset murrokset yleensä nostavat kysymyksiä siitä kuinka kyseistä asiaa voitaisiin hyödyntää liiketoiminnan tarpeisiin.

Tämä tutkimus tarkastelee tekoälyn mahdollisuuksia yritysanalytiikassa. Tavoitteena on tutkia, pystytäänkö laajoista teksteistä luomaan yhteenvetoja sekä analysoimaan tunnetiloja. Tutkimuksessa käytetään keinotekoista dataa. Toinen tavoite on rakentaa prototyyppitason ohjelmistoratkaisu, jolla näitä kyvykkyyksiä voidaan todentaa.

Microsoftin tarjoama referenssitoteutus tarjosi nopean tavan päästä eteenpäin ja keskittyä varsinaisiin tutkimuskysymyksiin sen sijaan että rakennettaisiin perustoiminnallisuutta. Ohjelmistossa käytettiin niin kutsuttua RAG-mallia (Retrieval Augmented Generation). Tämä mahdollisti oman datan ja valmiiksi koulutetun tekoälyn kyvykkyyksien yhdistämisen. Tutkimuksessa hyödynnettiin Design Science Research (DSR) metodia muistuttavaa työtapaa sekä ohjelmiston iteratiivisessa rakentamisessa että tekoälyn mahdollisuuksien selvittämisessä. Tarinankerrontaa hyödyntävää lähestymistapaa käytettiin DSR-prosessin ymmärtämiseksi syvällisemmin.

Tutkimus osoittaa, että tekoäly pystyy luomaan oikeanlaisia yhteenvetoja tekstistä ja tunnistamaan kohtuullisen hyvin myös tunnetiloja. Sekä rekursiota että tekstien pilkkomista osiin käytettiin hyödyksi, jotta voitiin välttää konteksti-ikkunan merkkirajoitteet. Tehokkuuden parantamiseksi käytettiin rinnakkaisuutta rajapintakutsujen yhteydessä. Työssä esiteltiin muutama uusi konsepti, jotta paloissa käsittely ja riittävä kontekstitietoisuus voitiin toteuttaa. Yksi näistä konsepteista nimettiin ”peruskysymykseksi” (engl. ”base question”). DSR-metodin osalta tehtiin joitain havaintoja, ja ehdotettiin esimerkiksi ketterien kehitysmenetelmien näkökulman lisäämistä prosessiin.

Yksi ehdotetuista tulevaisuuden tutkimuskohteista on peruskysymyksen idean kehittäminen. Lisäksi ehdotetaan tutkimaan voisiko laajemman tekstiaineiston alkuosan ylikorostusta yhteenvedoissa lieventää, ja voisivatko tekoälyä ohjelmallisiin toimistotyökaluihin sisällyttävät yritykset lisätä mahdollisuuden käyttää muokattua järjestelmäkehotetta tietyissä käyttötapauksissa tekoälyn ”käyttäytymisen” muokkaamiseksi. Käytännön toimenpiteenä ehdotetaan pitäytymistä kaupallisissa työkaluissa erittäin nopean kehitystahdin vuoksi.

Avainsanat: Tekoäly, Retrieval Augmented Generation, Design Science Research, tekstin yhteenveto, tunnetila-analyysi, peruskysymys

PREFACE

I am almost finished – but fortunately in good way! Studying alongside the daily consulting work has been anything but easy. That translates as “late nights” in English, for more than three past years already. However, I am very happy that I am about to reach the last milestone that also signifies graduation. It has all been worth it.

I would like to express my gratitude and thanks to everyone who has been supporting me along the way. This means my family, relatives, friends, teachers, and my employer Sininen Polku. In addition, I would like to separately thank professor Pekka Abrahamsson for enthusiastic attitude and great support and guidance during the process of making this thesis a reality. Further, great thanks to you Harri and colleagues for sponsoring me with the environment and providing very interesting research questions to study.

Last, but not least I would like to thank the Light of my life! You know who you are.

Kaarina, 11.01.2024

Henri Heimonen

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ADSRM	Agile Design Science Research Model
AI	Artificial Intelligence
AIGO	OECD Expert Group on Artificial Intelligence
a.k.a.	Also known as
API	Application Programming Interface
AWS	Amazon Web Services
BI	Business Intelligence
DB	Database
DSR	Design Science Research
DSR-IS	Design Science Research in IS
GPT	Generative Pre-trained Transformer
ICT	Information and Communication Technology
IS	Information Systems
ISDT	Information Systems Design Theory
IT	Information Technology
LLM	Large Language Model
LM	Language Model
OECD	Organization for Economic Co-operation and Development
PDF	Portable Document Format
POC	Proof-of-Concept, early prototype to understand feasibility
Q&A	Questions and Answers
RAG	Retrieval Augmented Generation
sci-fi	Science-Fiction, specific genre on movies
SW	Software, computer program
SQL	Structured Query Language
UI	User Interface

1. INTRODUCTION

As a field of study, artificial intelligence (AI) has existed for several decades, longer than one might anticipate. First actual work that is considered to concern artificial intelligence is already from year 1943 (Russell and Norvig, 2016, p. 16). However, for public audience the artificial intelligence, or AI for short, has primarily been the subject of sci-fi movies, or distant academic topic, for quite some time. During the past few years there has been a growing number of various enhancements or additions to existing everyday applications attributed to the use of “AI”, “Machine Learning” and so forth. Examples familiar to many are automatic enhancements and filters to photographs taken with the smartphones and shared through social media. Voice-controlled virtual assistants like Apple’s Siri or Amazon Alexa allow users to give commands and instructions by speech. Still, despite of many similar examples, the AI mostly stayed on the level the curiosity, something that was there to impress for a while but nothing really lifechanging. Until recently.

The status quo appeared to radically change soon after the launch of ChatGPT 30th November 2022 (Schulman *et al.*, 2022). On that day US-based company OpenAI announced their AI model and enabled people to have a dialogue with the artificial intelligence using the web interface. Soon after, the use of ChatGPT started to cause a lot of buzz that became visible even to someone not actively following the field. Bommasani *et al.* (2023) even used the characterization like “Models like OpenAI’s Chat-GPT have taken the world by storm”. Since then, large amount of news articles and blog posts have been published on ChatGPT in particular, but also on more generic level about the possibilities, pitfalls, and even threats posed by the artificial intelligence. Apprehension following the sudden advancements was not only the opinion of the public unfamiliar with the field, but also shared by several known and influential players in the field of artificial intelligence. They announced the concerns in an open letter, signed by more than 33 thousand individuals (“Pause Giant AI Experiments,” 2023).

As an outcome of the recent events, AI was no more just a curiosity or a topic of yet another movie, but something that was seen to change many things even in near future. While the interest of public was awakened, the same happened within organizations and industry. Suddenly there was a lot of talk about using ChatGPT for various applications, code generation, and so on. While some still wanted to put emphasis to obvious shortcomings or potentially false information created (“hallucinated”) by generative AI, others believed this would define the decade 2020’s.

1.1 Background for the Study: too much data

Sherman (2014, p.4) talks about the "data deluge". In other words, there simply tends to be more data than enterprises can handle, and the volume and variety keeps on growing constantly. Data, however, is the source of information, which in turn is the source of knowledge – if used properly. Sherman describes the knowledge as "the lifeblood of a thriving enterprise". (Sherman, 2014, p. 4). Data therefore is very valuable starting point, but there just often is too much of it and situation becomes overwhelming. This is the dilemma that industry aims to solve in various ways, to make sense out of the data. It is the chaos that must be taken under control.

On high level the role of the entire business intelligence domain, BI, is to extract actionable information from data. This means information that business can use and act on. Too much data, i.e. the above-mentioned "data deluge", however, can yield into inability to analyze the data, or alternatively lack of current enough information that would be useful. (Sherman, 2014, p. 10). There are numerous of tools and alternative practices available within the entire BI scene, but building full-blown enterprise-level data warehouse for instance may not be the viable solution for smaller companies.

This thesis is built around the business proposition of a small company based in Finland, that is interested to utilize the AI to support their operations and to build the competitive advantage. One of the possibilities of generative AI is to make sense of, and generate, textual data. The specific need this company has is to extract various actionable insights from textual data that is being generated in various contexts through their customers. Depending on the scenario, the amount of data rows can reach up to tens of thousands per use case, according to company representative. This may not sound much to someone working with relational databases, but in this case the data is actual text written by humans, and the quality of text varies a lot.

Therefore, no actionable information can be carved out by just making a SQL query. Extracting the insights requires going through the textual data and comprehension of actual content. As this has been currently done completely manually, the challenge goes back to the "data deluge" scenario in its own specific way – there is too much data to handle without spending a lot of time with that. In this case "too much" is not about bits or kilobytes. Instead, the challenge comes from the nature of data: analyzing the meaning of a sentence is not an arithmetic operation, where computers are usually good at. Rather, it requires semantics, something which has not been the strong aspect of computers in the past.

1.2 Research Questions

This thesis is built around a primary high-level research question:

- **How to use Large Language Models (LLM) effectively in complex business analytics?**

However, business analytics is a vast domain consisting of various topics and technologies. In addition, there is a particular angle to this research as explained in previous subchapter. Therefore, the scope and complexity will be effectively reduced to a case study, by introducing three sub-questions. They relate to the primary high-level research question while focusing to actual business scenario, handling the human-generated textual data. Three sub-questions are introduced next:

- **Q1: How can artificial intelligence be used to create summaries and extract actionable insight from textual data?**

The first research question mostly concerns the feasibility of the overarching idea of using the AI instead of manual human labor to comprehend the given input. In this specific case, company management needs to see what the overall summary is, related to the data collected from certain context, but without spending hours to go through the data line by line. Another important goal is to understand if AI can extract the actionable insights from the textual data. These insights can be improvement ideas related to specific context or domain for instance. It needs to be studied if the maturity of the technology is on the level where this is possible with reasonable outcome.

There are specific subtopics related to the feasibility of using AI and Large Language Models to process the large textual documents. For instance, if LLMs are used, how does the limitation of the prompt size affect the feasibility? Prompt can be described as an input that has the question to answer, added with guidelines to the operation of LLM model (behavior), but it also includes the output as model aims to complete the prompt (i.e. to answer the question posed). All this eats up the context window.

Other subtopic for the feasibility is about understanding how the AI can be guided to respond questions using the company data. The datasets are changing and proprietary, and naturally have not been available for AI during the training.

- **Q2: How can AI be used to detect sentiments from the text?**

For the given background, one important angle for the analysis is to understand the various sentiments of given input. This relates to the aspects of overall atmosphere or attitude buried within the data. This in turn may yield into actions, depending on the results and scenario itself. As for Q1, studying this question is also about understanding the technical capabilities: is this something where AI can help, or something that still should be done manually by human?

One of the subtopics related to sentiment analyses is the language used in the datasets. In the current case feedback can be either in English or in Finnish. If the AI to be used does not include native support for Finnish, what happens to sensitivity of sentiment analysis if dataset needs to be translated e.g., into English?

- **Q3: What kind of technical architecture and implementation would answer the need?**

Last sub-question relates to former two research questions as a logical step: as we aim to study the use of AI in data analysis, what kind of architecture can we use? In other words, what kind of elements or building blocks would be needed and how would they interact with each other? For the implementation part the goal is to create a Proof-of-Concept (POC), which is an early prototype-version software that demonstrates the essential required capabilities and their feasibility. POC-level software implementation is not productized solution, but indication whether the proposed approach and ideas are feasible or not. Actual productized software is scoped out of the thesis.

1.3 Methodology

This study focuses first on theoretical background through the literature. Then, concept of DSR in opened in more detail in its own dedicated chapter “3. Design Science Research”. Following that, technical implementation is studied and proposed to solve the research questions. This proposal is realized through the creation of an early prototype version of the software, or POC, to demonstrate the feasibility.

Through iterations, answers to research questions are constructed in iterative manner. In scenarios like this where various artifacts are essential pieces of solution the application of Design Science Research (DSR) method is useful approach.

1.4 Outcome

The outcome of the study is that AI can indeed be a valuable help in analyzing the large quantity of textual data, summarizing the contents, and extracting valuable insights out of it. However, due to perceived rapid advancements and regular announcement of new capabilities, the recommendation to use the commercially available (or soon to be available) tools is made.

Through the iterations, few technical alternative solutions were studied to gather and increase understanding related to research questions. As a practical deliverable, code repository for POC exists and will be available for the company. Essential parts of the program code are depicted and detailed out in this documentation. During the thesis detailed view to DSR methodology itself was taken as well, resulting in few observations.

Despite of the mentioned recommendation to stick with the commercial tools, this study has provided additional understanding for the cases where custom solution would be needed, or if commercial tools don’t work well for a specific use case. As a part of research, few new concepts were introduced. These innovations focus on handling large texts in parts. This would be required when context window size of available AI is limited and requires splitting the input data. It was noticed that when handling partial data there are pitfalls where AI can get confused and give unexpected results. Especially a new concept called “base question” – a way of “relaxing” user’s prompt – could be useful in such cases.

Finally, it can be mentioned that currently the pace of the development around AI seems incredibly rapid, new announcements are constantly coming. More tools get deployed, and today’s issues get “automatically” solved by those. Hence it is recognized that this work has delivered a kick-off towards the adoption of AI but will not remain fresh for long in terms of technical merits. However, lot of useful insights for the use of ChatGPT as a component of software has been gathered, and areas like prompt engineering skills can be expected to be useful also in future if using LLM programmatically.

2. PROBLEM STATEMENT AND THEORETHICAL BACKGROUND

The problem statement can be traced back to initial questions from the company associated with the thesis: how could AI be used to replace some labor-intensive phases in drawing insights from textual data? Can it help with analysis and summaries? Could sentiments be detected using AI? What kind of software architecture could be used? The last question about the POC architecture and implementation is somewhat separate from other research questions. In an analogy, it is a question about what kind of construction methods and elements should be used to build a house, whereas other questions are more about finding the feasibility of the ideas regarding the house to be built – if it is to be built. To be able to approach these questions few concepts will be explained first.

2.1 Business Intelligence (Business Analytics)

Howson describes business intelligence being “a window to the dynamics of a business”. (Howson, 2014). In other words, through business intelligence people in the organization can observe topics like performance, analyze operational efficiency, or perhaps discover new opportunities. Howson defines the business intelligence as “a set of technologies and processes that allow people at all levels of an organization to access and analyze data”. (Howson, 2014).

To open the concepts related to business intelligence Sherman uses familiar analogies from cooking in his business intelligence guidebook: he defines data as “a collection of ingredients”, for instance various vegetables from the market, some chicken, rice and so on, that are needed to begin the process of cooking. (Sherman, 2014, p. 9). According to him the information, on the other hand, is when these ingredients have been washed, peeled, cut into pieces and boiled into soup (Sherman, 2014, p. 10). However, having the soup ready does not help until it is served and eaten. This final step reflects gaining knowledge by actually consuming the information that has been made available – eating the soup, so to speak. (Sherman, 2014, p. 10).

Howson’s idea adds on this when she says that “Without people to interpret the information and act on it, business intelligence achieves nothing.” (Howson, 2014). Interpreting is about gaining the knowledge, but even after that there is the last step to take: acting, based on acquired knowledge. This is relevant point also to this research, as even though the aim is to use the AI to extract the information, the last step, acting, is still up to people. We are not putting the AI in change of business decisions – at least not now.

When talking about the semantics, in her book Howson states that the meaning of term “business intelligence” varies to different people. Further, some try to make a difference between “business intelligence” and “business analytics”, claiming that the former refers to historical and simple reporting, whereas the latter includes more advanced analytics aspects, like predictive analytics. (Howson, 2014). Howson disagrees with this, however, and states that use of these terms is mixed. Regardless of the exact terminology, the data and what can be derived from it should always be at the core. (Howson, 2014).

In this thesis we use terms “business intelligence” and “business analytics” interchangeably, referring to the same thing: data is collected, turned into information, and consumed to gain knowledge and actionable insights.

2.2 Artificial Intelligence

Artificial intelligence has already been mentioned several times until now in this thesis, but it hasn’t yet been properly defined for readers. Term “AI” may mean different things to different people – just like “business intelligence” can. The OECD even states that “There is no universally accepted definition of AI.” (OECD, 2019, p. 22).

Over the years several definitions for artificial intelligence have been created by scholars. For instance, Haugeland talks about “The exiting new effort to make computers to think” and even uses words “machines with minds”. (Haugeland, 1985, according to Russell and Norvig, 2016, p. 2). Winston’s definition for AI is “The study of the computations that make it possible to perceive, reason, and act.”, even though he also states that there are many possible ways to define it, and this is one of them. (Winston, 1992, p. 5).

Poole *et al.* prefer to talk about “computational intelligence” over the “artificial intelligence” as they see the term “artificial” rather confusing, but they mean the same field, nevertheless. (Poole *et al.*, 1998, p. 1). They state that “Computational intelligence is the study of the design of intelligent agents.” (Poole *et al.*, 1998, p. 1). Agent does not refer to people with black suits and glasses, but basically anything that interacts with its environment. Examples of agents they give include dogs, thermostats, humans and so on. What then makes agent as intelligent is that its actions are appropriate against its circumstances and goal. They also expect intelligent agent to be able to adapt to changes in the environment and goals, and to learn from experiences. Intelligent agent, they say, “makes appropriate choices given perceptual limitations and finite computation.”. (Poole *et al.*, 1998, p. 1).

Continuing with the various definitions for AI, Rich *et al.* define AI as following: "Artificial intelligence (AI) is the study of how to make computers do things which, at the moment, people do better." (Rich *et al.*, 2010). This definition appears a bit ambiguous, which authors also agree themselves. They however believe this definition can avoid certain arguments that usually arise from the meaning of terms "artificial" or "intelligence".

Russell and Norvig (2016, p. 2) have organized several definitions of AI (including most of the above-mentioned ones in this) into two domains: 1) thought process and reasoning, and 2) behavior. They use these two dimensions to indicate how the primary perspective of various definitions for AI could be split: some focus on thinking, others more on acting.

In addition of these dimensions, Russell and Norvig introduce two criteria which indicate how various definitions of AI suggest the success should be measured: is the performance of AI compared to human performance, or rather into ideal performance, which they call "rationality". (Russell and Norvig, 2016, p. 1). The outcome is four categories for organizing the AI definitions. (Russell and Norvig, 2016, p. 2). The idea is depicted in the following Figure 1, adapted, and simplified from their work:

Thinking Humanly "The exciting new effort to make computers think… *machines with minds, …"* (Haugeland, 1985)	**Thinking Rationally** "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly "Artificial intelligence (AI) is the study of how to make computers do things which, at the moment, people do better." (Rich *et al*., 2010)	**Acting Rationally** "Computational intelligence is the study of the design of intelligent agents." (Poole *et al*., 1998*)*

Figure 1. *Some definitions of artificial intelligence (adapted from Russell and Norvig, 2016, p. 2).*

Regarding the separation between human and rational, Russell and Norvig emphasize that they are not suggesting human behavior to be irrational or insane per se but are noting there are imperfections in human reasoning (Russell and Norvig, 2016, p. 2). They refer to cataloged errors in human reasoning (Kahneman *et al.*, 1982, according to Russell and Norvig, 2016, p. 2).

Tversky and Kahneman (1974) studied various beliefs and biases impacting human reasoning and decisions, first published in their impactful article in Science Magazine. One example of the systemic error proposed by Tversky and Kahneman is so called “representativeness”, in which people use the likeness as a guiding heuristic and tend to ignore for instance statistical probability. (Tversky and Kahneman, 1974). Another example they give in the article is ”availability” bias, where probability or frequency of something is assessed based on how easily similar class or event can be remembered. These and similar systemic errors can make human reasoning erroneous in certain scenarios, hence it is useful to refer rather to ideal behavior or reasoning when defining AI.

According to Russell and Norvig the approach around rational agents (i.e., “Acting Rationally” category) has couple of advantages over other categories (Russell and Norvig, 2016, p. 4). First, they comment it being more generic than “Thinking Rationally” category because correct inference, emphasized in rational thinking, is “just one of several possible mechanisms for achieving rationality”. (Russell and Norvig, 2016, p. 4). Sometimes the correct thing cannot be proved fully by logic, but still action needs to be taken, regardless of limited inference possibilities. In some situations, like avoiding serious incident, more reflex-like actions are better than slow deliberate thinking, followed by an action (Russell and Norvig, 2016, p. 4). Still, according to OECD the definition of “The study of the computations...” (Winston, 1992, according to OCED, 2019, p. 22), is aligned with the definition used by Russell and Norvig. (OECD, 2019, p. 22).

As a second advantage of “Acting Rationally” category across others, Russell and Norvig add that “The standard of rationality is mathematically well defined and completely general”, something which according to them cannot be said from human behavior (Russell and Norvig, 2016, p. 5). To align to human behavior the specific environment should be adapted for, and somehow the definition would have to be derived from “the sum total of all the things that humans do.” (Russell and Norvig, 2016, p. 5).

Above-mentioned two reasons are why Russell and Norvig propose to use the approach of “rational agents” as a preferred way for defining the AI. (Russell and Norvig, 2016, p. 5). The definition of rational agents aligns with the definition by Poole *et al.,* even though the term they use is “intelligent agents” (Poole *et al.*, 1998, p. 1).

A high-level conceptual view of AI, as proposed by the OECD Expert Group of Artificial Intelligence (AIGO) if depicted in the following Figure 2:

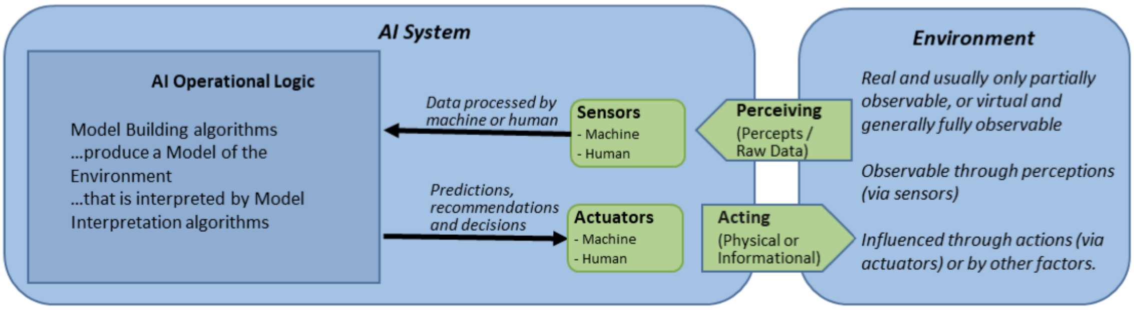


Figure 2. *A high-level conceptual view of an AI system (AIGO, 2019, p. 6).*

AIGO states that the entity labelled as AI System in the conceptual presentation is “also referred to as ‘Intelligent agent’”. (AIGO, 2019, p. 6). Main three elements of AI System are listed in the following (AIGO, 2019, p. 6):

- Sensors – for collecting raw data from the Environment.
- Actuators – for changing the state of Environment by taking actions.
- Operational Logic – for providing instructions for the Actuators, based on data from Sensors, guided by given objectives.

Depending on one’s background, the concept of “Sensors” or “Actuators” might lead to the direction of some tangible, physical pieces of hardware. But we should not be limited by this, even thought, for instance in the case of robots, this might be the case. However, in case of software agent, for instance, Russell and Norvig give other kind of examples: SW agent might receive for instance keystrokes or data from files as an input, and then influence its environment by showing something on the screen or perhaps sending data to the network, and so on. (Russell and Norvig, 2016, p. 34).

Hence the AI, or intelligent/rational agent, is something that interacts with its environment in a rational way. The rationality, or intelligence, of the AI system is measured against the defined goals and criteria instead of concept of “intelligence” just existing in a vacuum as an absolute concept. Desired behavior, and hence intelligence way of acting at given scenario is guided by the targets set. In addition to performance measure, other factors that influence on what is rational at certain point is agent’s earlier knowledge of the environment, actions that agent could perform, and finally its percept sequence, according to Russell and Norvig. (Russell and Norvig, 2016, p. 37). Percept sequence means basically everything that agent has ever perceived to date via inputs, but usually limits are being set how long sequence makes sense to consider, when making choices. (Russell and Norvig, 2016, pp. 34–35).

Using the above four factors (performance measure, prior knowledge, available actions, and percept sequence) Russell and Norvig settle with the definition of a rational agent: "*For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.".* (Russell and Norvig, 2016, p. 37).

2.3 Large Language Models

As already claimed earlier, launch of ChatGPT was a game changer in a sense that it brought AI and especially Large Language Models heavily into public discussion. There are predictions how the nature of the labor at certain fields will be drastically altered in coming years. In addition, there are enthusiastic promoters for the AI, but also those who keep warning about threats, probable or imaginary.

But what is LLM, after all? Is it a powerful, conscious electronic mind, perhaps a mere guessing machine putting words together, one after another, or something else? What about the relation of AI and LLM, are they the same or different? Following subchapters will study these questions about the LLM, also because the Proof-of-Concept SW created as part of the thesis utilizes one LLM implementation.

2.3.1 Language Models

If language model is said to be "large", the question to make even before diving more into definition of LLM might be that what's the mere "language model" (LM), without being large? Eovito and Danilevsky write that "*A language model* is a technique for calculating the probability of a particular sequence of words occurring." (Eovito and Danilevsky, 2021). According to them LM aims to imitate human "linguistic capabilities, learning a myriad of associations between words to represent". (Eovito and Danilevsky, 2021). To emphasize the difference between imitating and really knowing, however, they use the example: like humans don't really know how it is like to be some animal, like a bat for instance, likewise the LM does not know what it is to be a human – no matter how believable the interaction. This remains the same, regardless how much we would try to imitate bat's behavior. (Eovito and Danilevsky, 2021).

The question whether human could understand what it's like to be a bat is borrowed from the philosopher Nagel by Eovito and Danilevsky: Nagel came to conclusion that we wouldn't know how it's to be a bat, we would only at best understand what it would be for us "to behave as a bat behaves." (Nagel, 1974, according to Eovito and Danilevsky, 2021). Similarly, the language model can imitate us, but it does not know what it's like to be a human. (Eovito and Danilevsky, 2021).

At this point the definition of LM by Eovito and Danilevsky sounds a lot like above-mentioned guessing machine. Where is the intelligence? However, as we have shown earlier, in case of AI the "intelligence" is not about possessing a conscious mind, but more about behaving in rational way according to the definition of "intelligent agent". Putting it differently, rational or intelligent behavior means doing the right thing, guided by the defined goals and performance metrics in certain environment.

To emphasize the fact that there is no conscious mind behind LM, Eovito and Danilevsky say: "Language models do not mean what they say. Language models generate well-formed language, and humans experience it as an illusion of meaning." (Eovito and Danilevsky, 2021).

When looking for other definitions, Yogatama *et al*. refer language models as "artificial language processing systems" in their article. (Yogatama *et al.*, 2021). To take the simplification metaphor even further, Bommasani *et al*. state that "At its core, an LM is a box that takes in text and generates text". (Bommasani *et al.*, 2023). Following Figure 3 depicts the idea:

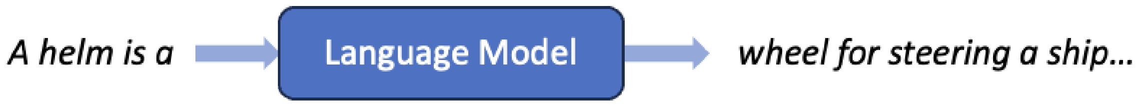


Figure 3. *Language model (adapted from Bommasani et al., 2023).*

Eovito and Danilevsky describe the operation of LM by comparing it to a game where a player gives a word, then next one gives the next word to continue the sentence, and so on. LM does the same, it takes the prompt (which could be e.g. "A helm is a" from the previous Figure 3) and plays the next round of the game of "predict the next word", by outputting that word. This "prompt" can be a sentence that needs to be continued, or something we want to translate. The goal is to output the best possible word (or words). What is considered as best, depends on the training of the model. There are three generic ways how language models represent languages: defining a set of rules (linguistic approach), relying on statistical probabilities, or using so called "Embeddings" which defines the vector representation for each word in multi-dimensional space. The last of these is used mostly in state-of-the-art, deep neural network language models, according to Eovito and Danilevsky. (Eovito and Danilevsky, 2021).

Combining all the above together we can state that LM is one category or implementation of AI. LM is the AI model, designed to complete the given prompt in the best possible way, guided by its training.

2.3.2 Large Language Model vs Language Model

What about the Large Language Models, then? In their work Rae *et al*. refer to the language models that belong to the of class of neural networks called as “Transformers” (Vaswani *et al.*, according to Rae *et al.*, 2022). They continue by saying that “There has been a trend of scaling the combination of training data, model size (measured in parameters) and training computation to obtain models with improved performance across academic and industrial benchmarks.” (Rae *et al.*, 2022). A list of examples including GTP-2 with 1.5 billion parameters, and GPT-3 with 175 billion parameters, are given. They finally state that “The moniker *Large Language Models* (LLMs) has become popular to describe this generation of larger models.” (Rae *et al.*, 2022).

In other words, Large Language Model is a type of Language Model, characterized by notably large dataset and number of parameters. Therefore, all the generic explanation about Language Models above applies to LLMs as well.

2.3.3 Tokens

One of the terms that comes up when discussing Large Language Models is the concept of “token”. Token is simply a “text fragment”, when considering textual data. (Egli, 2023). However, token is even more generic concept and can mean a fragment of image, textual or audio data as well (Child *et al.*, 2019). In the scope of the thesis, however, the idea of text fragment is feasible.

Earlier the operation of Language Models was compared to word prediction game, but another description could be “predict the next token”. Like stated by OpenAI documentation, “Language models read and write text in chunks called tokens.” (“Text generation models,” n.d.). Tokens are language specific and can range in length from even shorter than one character to longer than one word. In English the range is between one character and one word. With ChatGPT the number of total tokens consumed impacts e.g. the API (Application Programming Interface) call cost as billing is per token. Also, there is a total limit of tokens, which must not be exceeded for the API to work at all. (“Text generation models,” n.d.).

This token limit is one of the essential dilemmas dealt with the POC software implementation – how to handle documents that exceed the available window size? However, as new models get introduced the size of the available context window keeps on expanding.

2.3.4 Prompt

We already referred to the concept of “prompt” earlier when explaining the Language Models. There are couple of layers related to concept of “prompt” that are relevant to our case. Like already stated earlier, prompt is the sequence of words (or tokens) that Language Model aims to complete. That is the part visible to end user, direct and immediate.

There is also another, more hidden aspect of prompt, so called “system” prompt. With ChatGPT a related system message can be used to guide the persona, or behavior of the model. (“Prompt engineering,” n.d.). This is essential part of the implementation of POC software. We call this part of the prompt as “system prompt”, same term is also available elsewhere on the internet. This system prompt is relevant to us, for guiding the behavior of an AI model, but also because it is sent to the model and consumes the tokens similarly as the visible prompt. (“Text generation models,” n.d.).

2.4 Conceptual Setting

All the previous concepts have been explained as they come together in the conceptual setting related to this study. The simplified targeted operating environment setting, that abstracts many details away, is shown in the following Figure 4. The parts directly related to this thesis are highlighted with blue color.

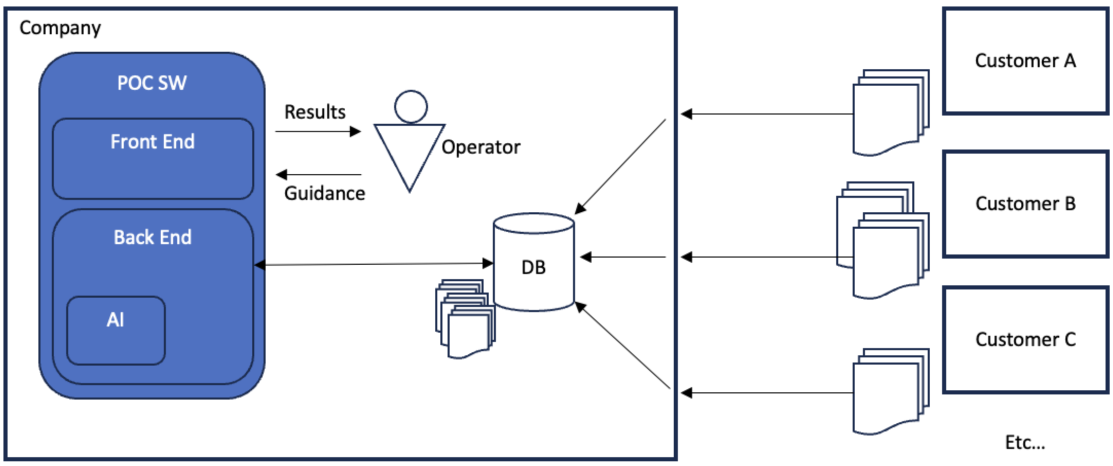


Figure 4. *Target operating environment. Scope related to thesis highlighted.*

Company associated with the thesis has several customers, all delivering the textual input data from various scenarios which are not opened in detail here to retain necessary confidentiality. Company stores their customers' data securely in cloud storage. After receiving the input data from customer, the next step in current setup is to take the data under further manual analysis and draw conclusions from there. This is the business analytics part, essential to the process.

This study aims to change things, however. In the revised setup described in Figure 4, the company employee ("Operator") uses the POC software to orchestrate the entire analysis process. He/she communicates with the software UI ("Front End"), through which also the results will be delivered. Communication happens by giving prompts through the chat window, using natural language, and results get output to the same user interface. Result can be for instance a summary of handled document, or insights based on direct question, for instance listing most promising improvement areas based on customer feedback. POC software "Back End" (non-UI related part of the software, logic) fetches the relevant data from the data storage as part of the process. The storage is simplified here as Database, DB, for simplicity. Back End also includes the communication with AI module and includes configuration of the hidden part of the prompt, i.e. system prompt. Due to the expected size of the documents, and earlier explained token limits, a solution that can work around these limitations is needed.

A reference implementation by Microsoft was used when building the POC software ("ChatGPT + Enterprise data with Azure OpenAI and Cognitive Search," 2023). Even though UI is part of the POC, very minimal changes to existing UI have been made as it already provides working user interface. Those changes are not shared in this document. Relevant parts of the Back End, however, went though some significant changes and these changes are opened later in detail.

2.4.1 Possible Strategies to Utilize Language Models

There are few possible ways to utilize the Language Models, the AI-part of operating environment. First approach would be to train the model from ground up, but this is very expensive and time-consuming and requires a lot of resources.

Another approach is to use pre-trained language model, what ChatGPT is for instance. This approach takes the model as-is, accepting its parameters and training as decided by whoever has trained the model. This is the approach used in this study and POC implementation and provides the fast track forward. Pre-trained model is given few examples via prompt which usually yields proper results already.

However, it is also possible to fine-tune the pre-trained model, which means customizing it for a specific application. According to OpenAI documentation, fine-tuning allows to achieve results with less examples, saving cost and latency (as examples and other parts of the ”system” prompt are always sent to API when calling for prompt completion). However, even fine-tuning is an investment requiring for time and effort. As an alternative before fine-tuning OpenAI recommends for instance “prompt engineering” and “prompt chaining (breaking complex tasks into multiple prompts)”. (“Fine-tuning,” n.d.).

2.5 Success Criteria

There are couple of ways to measure the success of the thesis: either via direct adoption of the POC software towards further productization, or alternatively by introducing generic impetus and ideas for pursuing AI-supported customer data handling and analysis.

Rate of change is currently so fast that that POC only gives certain basic ideas that can then be implemented with applicable technologies. Push towards adoption of AI in own business is already an achievement of its own, and this has already taken place in the company related to this study. Ultimately success will be measured by commercial results when question is observed from business perspective.

2.6 Most Essential Technical Building Blocks

Like mentioned, only minimal changes to the UI were made as that was not the focus of the study, it was mostly used as-is. UI will not be detailed further but can be referred to via available Microsoft’s GitHub repository. (“ChatGPT + Enterprise data with Azure OpenAI and Cognitive Search,” 2023). User Interface is how human operator communicates with the system, however, and hence integral part of the system.

The Back End part of the software contains the actual logic and is built with Python and several supporting libraries. In the final version of the software some technologies that were in the initial reference implementation were left out, simplifying the architecture. This is more explained in following chapters that describe the iterations of the system as it gets built up.

In the conceptual Figure 4 the AI has been drawn within the Back End. It could be represented also differently, however, as the actual AI resides outside of POC code itself. AI-part is implemented with Azure OpenAI API that communicates with the AI model. (“Azure OpenAI Service – Advanced Language Models | Microsoft Azure,” n.d.).

Azure Blob Storage is used for storing documents and backend-part of the software interfaces with it to fetch the documents for processing. (“Azure Blob Storage | Microsoft

Azure," n.d.). This is the "DB", database component of the system. However, it is merely used as a "dummy" component for the POC, that is why it is not specifically highlighted in the conceptual picture.

3. DESIGN SCIENCE RESEARCH

In their book, Vaishnavi and Kuechler emphasize the fitness of Design Science Research (DSR) method especially on Information and Communication Technology (ICT) field (Vaishnavi and Kuechler, 2015, p. 2). They state that the systems involving human-computer interaction are complex by nature and touch multiple fields of research. Therefore, questions without existing theoretical background may emerge. In these kinds of situations, they claim, “DSR—exploring by creating—excels.” (Vaishnavi and Kuechler, 2015, p. 2).

This chapter will open the concept of Design Science Research in detail, as approach close to DSR has been the primary methodology followed in this thesis (see chapter “Limitations”, however). Target of the thesis is to work on actual deliverables alongside the theory, therefore this kind of approach is quite natural.

Vaishnavi and Kuechler also describe the “Design Science Research in IS”, something they call as DSR-IS, which is less guided version of the DSR methodology in terms of requirements. (Vaishnavi and Kuechler, 2015, p. 60). DSR-IS requires three things that are “… construction of an artifact … , gathering of data on functional performance of the artifact (i.e., evaluation), and … reflection on the construction process and on the implications the gathered data … have for the artifact informing insights(s) or theory(s).”. (Vaishnavi and Kuechler, 2015, p. 60). As can be seen from the discussion later in this chapter where narrative approach is used, the actual approach applied for this study is closer to DSR-IS than pure DSR.

3.1 Basic Terminology

Vaishnavi and Kuechler define “research … as an *activity* that contributes to the *understanding* of a *phenomenon*.” (Kuhn, 1996, first published in 1962, and Lakatos, 1978, according to Vaishnavi and Kuechler, 2015, p. 9. They continue that phenomenon usually refers to “a *set of behaviors of some entity* (entities)”, and “*Understanding* in most western research communities is *valid (true) knowledge that may allow prediction* of the behavior of some aspect of the phenomenon. Thus, research must lead to contribution of knowledge — usually in the form of a theory — that is *new and valid (true)*.” (Vaishnavi and Kuechler, 2015, p. 9).

In their article handling the design science research methodology for information systems research, Peffers *et al*. define design as "the act of creating an explicitly applicable solution to a problem". (Peffers *et al.*, 2007). Vaishnavi and Kuechler state that "design deals with creating some new artifact that does not exist." (Vaishnavi and Kuechler, 2015, p. 10). Drawing from Walls *et al.*, Hevner *et al*. point out that "Design is both a process (set of activities) and a product (artifact)". (Walls *et al*., 1992, according to Hevner *et al.*, 2004). According to Vaishnavi and Kuechler the design can be "routine" if needed knowledge already is available, otherwise they say the design is "innovative". (Vaishnavi and Kuechler, 2015, p. 10).

Artifacts in IT (Information Technology) domain include "constructs, models, methods and instantiation" according to Hevner *et al*. (Hevner *et al.*, 2004). Orlikowski and Iacono (2001) describe wider scope of IT artifacts, suggesting to include for example historical and cultural aspects into conceptualization of IT artifacts.

Vaishnavi and Kuechler open the difference between Design Science and Design Science Research: they refer to Simon who talks about "natural science" and "science of the artificial", the latter also known as "design science". (Vaishnavi and Kuechler, 2015, p. 10; Simon, 1996). The former, natural science, refers to something occurring in nature or society, whereas the latter, design science, is "a body of knowledge about the design of artificial (man-made) objects and phenomena—artifacts—designed to meet certain desired goals". (Simon, 1996, according to Vaishnavi and Kuechler, 2015, pp. 10–11). Vaishnavi and Kuechler continue by defining the difference between Design Science and Design Science Research by stating that "*Design science* then is knowledge in the form of constructs, techniques and methods, models, well-developed theory for performing this mapping—the know-how for creating arti-facts that satisfy given sets of functional requirements. *DSR* is research that creates this type of missing knowledge using design, analysis, reflection, and abstraction." (Vaishnavi and Kuechler, 2015, p. 11).

Figure 5 below aims to summarize the earlier discussion and present the relations between the mentioned concepts, deriving from above-mentioned sources:

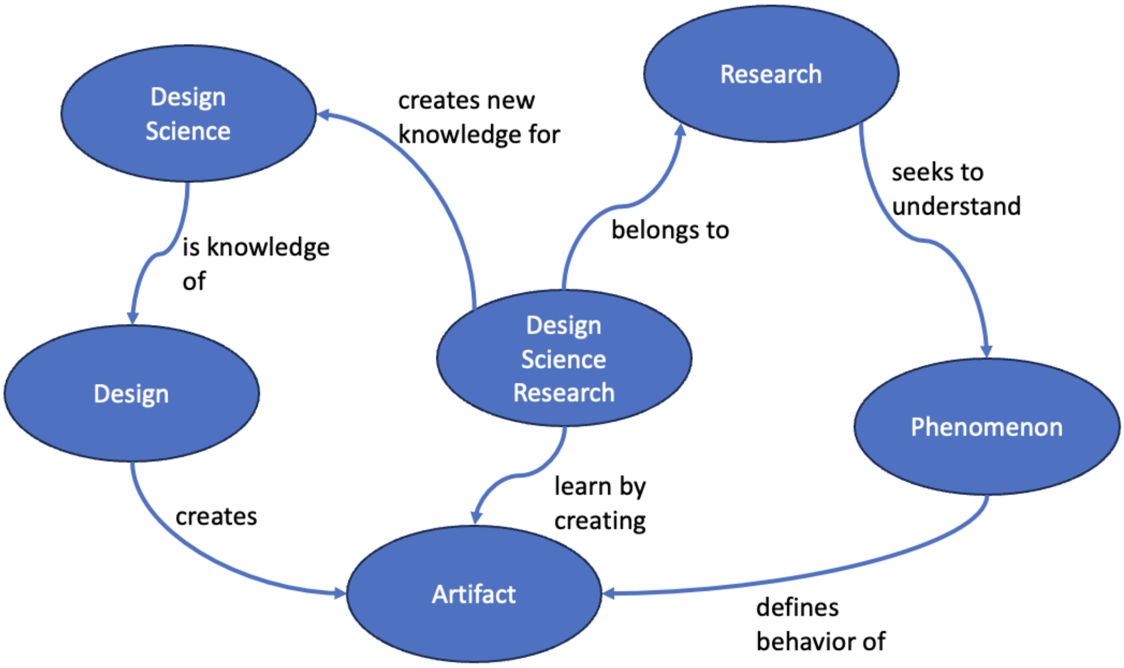
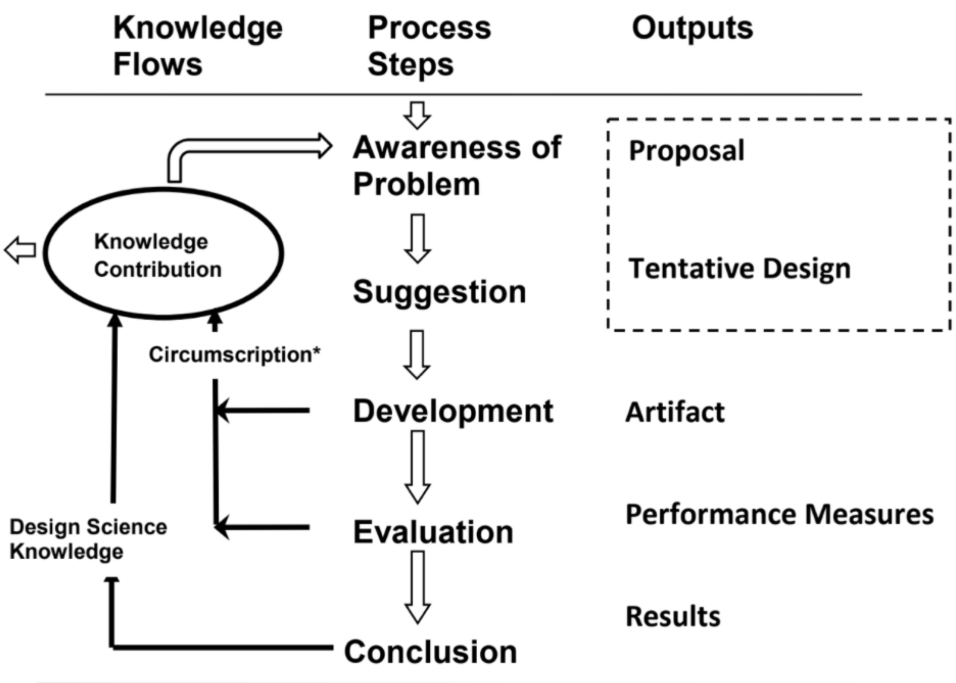


Figure 5. *DSR concepts and their interrelations.*

3.2 Design Science Research Process Model

In their article Hevner *et al.* say that "The fundamental principle of design-science research … is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artifact." (Hevner *et al.*, 2004, p. 82).

In essence the Design Science Research uses relatively pragmatic approach to solve research problems. Already in the early phases of the process model (or cycle, as the process is iterative) the aim is to create proposal and tentative design towards the question at hand. One version of the DSR process model is shown in Figure 6, as depicted by Vaishnavi and Kuechler (2004):



* Circumscription is discovery of constraint knowledge about theories gained through detection and analysis of contradictions when things do not work according to theory (McCarthy, 1980)

Figure 6. *Design Science Research Process Model (DSR Cycle) (Vaishnavi and Kuechler, 2004).*

As seen from the previous process model in Figure 6, the proposal is created as an output after becoming aware of the research problem. This proposal is more specifically a proposal to initiate research. Next phase is called as Suggestion phase, and it yields a tentative, initial design as an output, even though Vaishnavi and Kuechler mention that in more formal cases it is possible that it is included already in the proposal. (Vaishnavi and Kuechler, 2015, pp. 14–15). About the expected maturity of the design at this stage, Vaishnavi and Kuechler state the following: “Suggestion is essentially a creative step wherein new functionality is envisioned based on a novel configuration of either existing or new and existing elements.” (Vaishnavi and Kuechler, 2015, p. 15). This can be interpreted in a way that not that many details are yet expected.

Tentative design gets enhanced and implemented in Development phase. According to Vaishnavi and Kuechler “the novelty is primarily in the design, not the construction of the artifact.”, meaning that for instance the technologies or practices used in implementation are not the key, instead the design that aims to answer to research question is. (Vaishnavi and Kuechler, 2015, p. 16).

When artifact is available it is evaluated (Evaluation phase) against the criteria that has been set already in the proposal. Deviations from set expectations are to be explained tentatively. However, almost never in DSR things get completed here, but rather “the evaluation phase results, and additional information gained in the construction and running of the artifact are brought together and fed back to another round of suggestion”. (Vaishnavi and Kuechler, 2015, p. 16).

To confirm this thought, the Conclusion phase could signify the end of the research effort but also the end of a single research cycle only. Even in the final cycle there can be deviations from the anticipated (and revised) behavior of the artifact, but research effort can be finished if the results are found to be on satisfactory level. If cycle marks the end of the research effort the results are gathered and sometimes categorized as “firm” (facts learned or repeatable behavior identified) or “loose ends” (behavior that cannot be explained and can be topics for further research). (Vaishnavi and Kuechler, 2015, pp. 16–17).

As presented above, DSR output should produce new knowledge into design science domain. In their book Vaishnavi and Kuechler have adapted a simplified presentation out of article by Gregor and Hevner for possible types of output (Vaishnavi and Kuechler, 2015, p. 19). Original presentation by Gregor and Hevner is shown in the following Figure 7 (Gregor and Hevner, 2013):

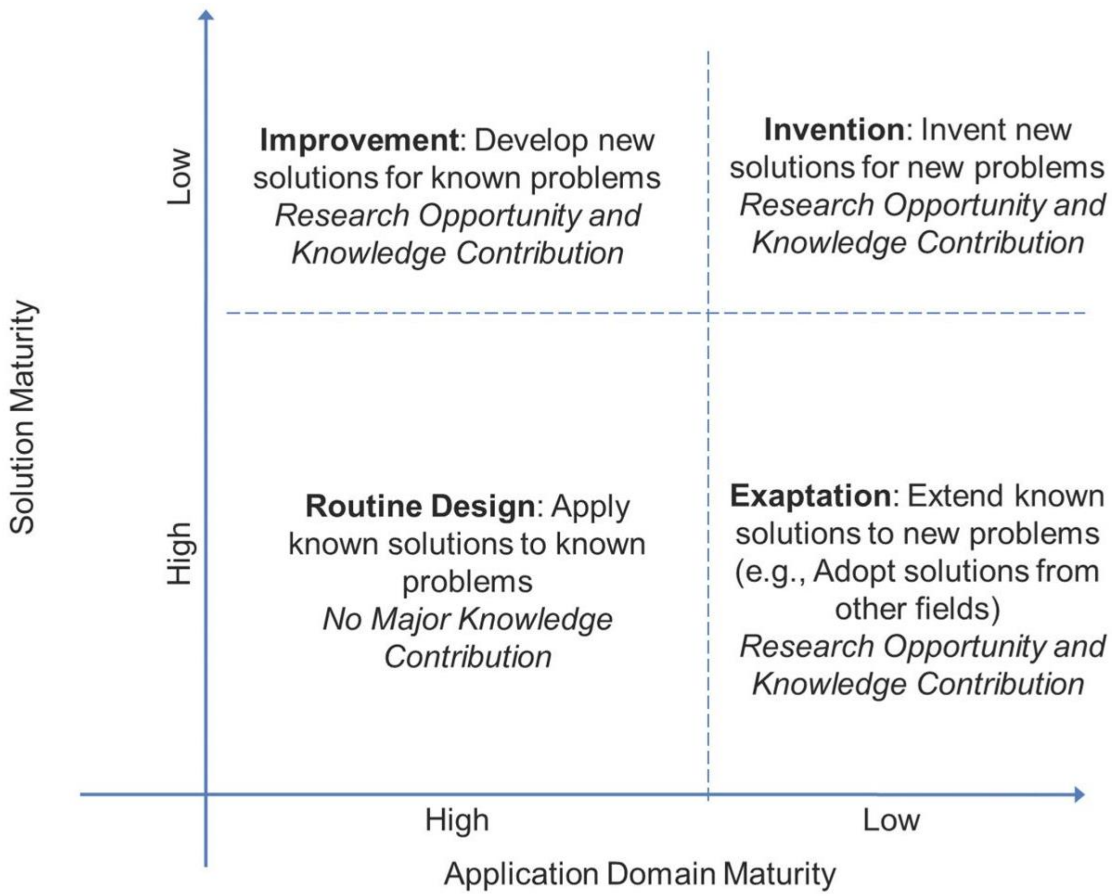


Figure 7. *DSR Knowledge Contribution Framework (Gregor and Hevner, 2013).*

Gregor and Hevner observe the contribution (DSR output) by utilizing two different axes to describe “Solution Maturity” and “Application Domain Maturity”. They also use term “problem maturity” as an alias for application domain maturity. These two domains describe the environment in which the DSR effort takes place. (Gregor and Hevner, 2013).

If the application domain (i.e., problem) maturity is low it means that “little current understanding of the problem context exists”. (Gregor and Hevner, 2013). They also state that in case of low problem maturity “…so little may be known about the problem that research questions may not even have been raised before”. (Gregor and Hevner, 2013).

Gregor and Hevner then use solution maturity axis to depict the maturity of existing artifacts, aimed to solve the problem at hand, or research question: “The y- axis represents the current maturity of artifacts that exist as potential starting points for solutions to the research question, also from high to low”. (Gregor and Hevner, 2013).

According to Gregor and Hevner, if the solution domain maturity is low (“…where no effective artifacts are available as solutions”), possible levels of contribution are either “improvement” or “invention”. (Gregor and Hevner, 2013). They define an “improvement” something brings new solutions to a problem that is known, whereas “invention” on the other hand does the same but for entirely new problems. They continue saying that if solution domain maturity is already high, expected levels of contribution are either “routine design”, or “exaptation” – latter of which Vaishnavi and Kuechler (2015, p. 19) have translated as “adaptation”. Routine design does not bring much new value in terms of knowledge contribution as it mostly applying existing artifacts to “to address the opportunity or question”. It operates on domain where much is already known, and available. Exaptation, or adaptation, extend knows solutions to new problems. (Gregor and Hevner, 2013).

Regarding the actual possible outputs from DSR project, Vaishnavi and Kuechler list the alternatives in the following Table 1, including short explanation of each (Vaishnavi and Kuechler, 2015, p. 20):

Table 1. *Potential Outputs of a Design Science Research Project (Vaishnavi and Kuechler, 2015, p. 20).*

	Output	*Description*
1	Constructs	The conceptual vocabulary of a domain
2	Models	Sets of propositions or statements expressing relationships between constructs
3	Frameworks	Real or conceptual guides to serve as support or guide
4	Architectures	High-level structures of systems
5	Design principles	Core principles and concepts to guide design
6	Methods	Sets of steps used to perform tasks—how-to knowledge
7	Instantiations	Situated implementations in certain environments that do or do not operationalize constructs, models, methods, and other abstract artifacts; in the latter case such knowledge remains tacit
8	Design theories	A prescriptive set of statements on how to do something to achieve a certain objective. A theory usually includes other abstract artifacts such as constructs, models, frameworks, architectures, design principles, and methods

Regarding the concept of “design theories” mentioned in the end of Table 1, the adaption of DSR for Information Systems, i.e., DSR-IS, rather talks about “information systems design theory (ISDT)”, initially suggested by Walls *et al*. (1992, 2004), according to Vaishnavi and Kuechler (2015, p. 59). ISDT “… is a set of primarily prescriptive statements describing how a class of artifacts should behave (meta-requirements) and how they can be constructed.”. (Walls *et al*., 1992, 2004, according to Vaishnavi and Kuechler, 2015, p. 61).

These potential outputs presented in Table 1 can also be further categorized, however. Vaishnavi and Kuechler adapt the work from Purao and present the previous list of outputs in few categories, defined by “level of abstraction and generalization” and continue by stating that “outputs at higher levels are preferred since it reflects a more general advancement of knowledge in the area” (Vaishnavi and Kuechler, 2015, p. 21). (Vaishnavi and Kuechler, 2015, pp. 21–22). Figure 8 lists these categories:

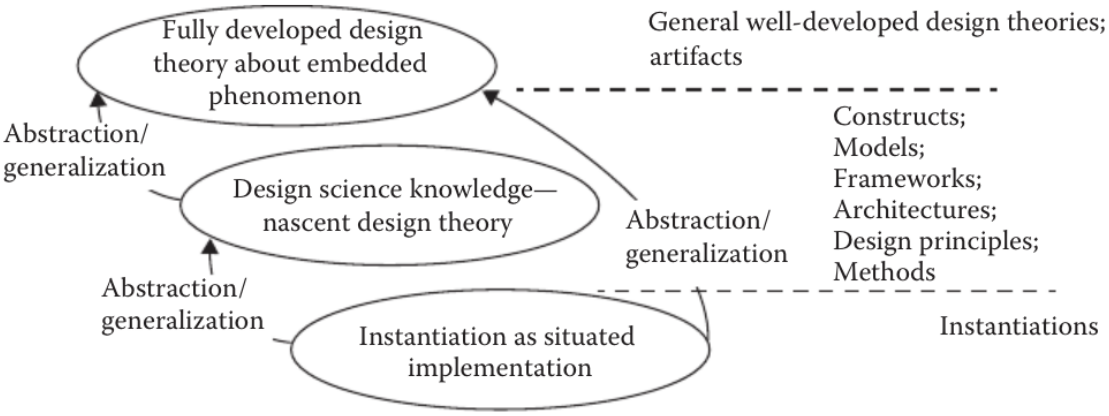


Figure 8. *Design science knowledge hierarchy. (Adapted from Purao, S., GSU CIS Dept. Working Paper, 2002, according to Vaishnavi and Kuechler, 2015, p. 22).*

If looking the DSR process cycle presented so far, it has certain "waterfall" type of tone in it, vs. the "Agile" approach that is quite dominant in software development nowadays. While assessing the DSR process via the narrative strategy (opened in next sub-chapter) also a proposal for agile-based DSR method was identified. This proposal brings in the concept of backlog, stories, demonstrations, attention to non-functional requirements and some other elements from IT agile ways of working as shown in Figure 9 (Conboy *et al.*, 2015):

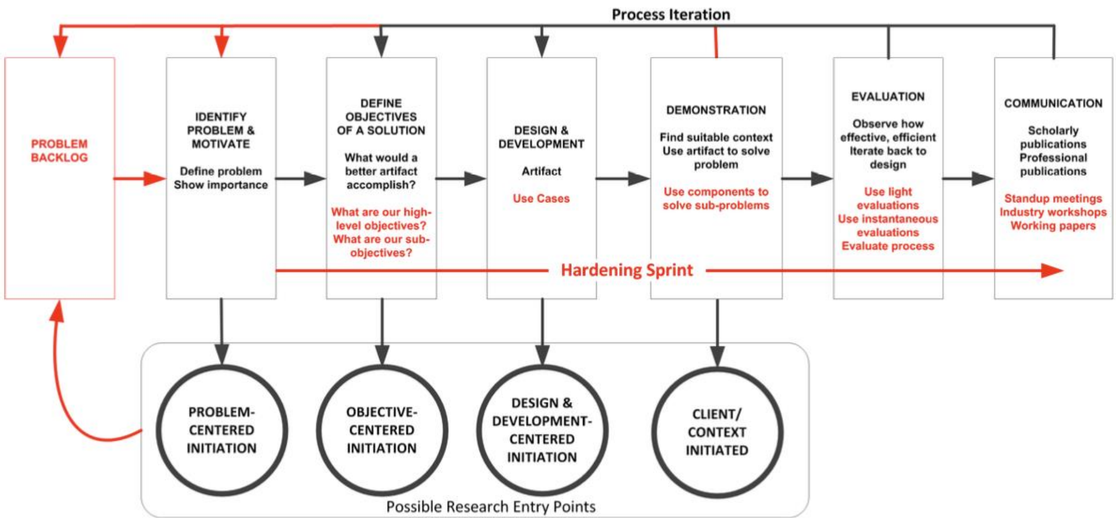


Figure 9. *Agile Design Science Research Model (ADSRM) (Conboy et al., 2015).*

3.3 Additional Ways for Perceiving the Design Science Research Process

To expand the understanding of the DSR process, the subset of strategies described by Langley in her article “Strategies for theorizing from process data” are used in this chapter. Langley studied various strategies that can be used to form theories from process data. In other words, how one can create generic process definition by observing and analyzing e.g. events and activities happening during the process. (Langley, 1999).

In this case the process under observation is the Design Science Research, and the process data hence are the events, actions and selections made during the iterative process when aiming to solve the defined research question and its derivatives (sub-questions).

The goal of using this additional approach here is an attempt to enrich the ways how DSR is perceived or described. In former sections the DSR process has been described by taking the definitions from literature and outlined as cyclic / iterative process with certain inputs and outputs, be them intermediate or final. Using additional strategies to understand the DSR is not going to change the nature of the core process but can open some new points of view.

One specific point of view to raise here is that as mentioned, process “close” to DSR was followed, but not exact process, by-the-book. Rather it was something closer to earlier-mentioned DSR-IS. This can be seen either as a strength, allowing additional perspective, or a weakness, undermining the comments given for DSR in first place. But quite often in real-life scenarios processes are guidelines that are there to support, but also could drive one into dead end if taken too literally. Usually, they must be adapted for the use case to give the support intended in the first place.

3.3.1 ”Seven Strategies of Sensemaking”

Langley’s “Seven Strategies for Sensemaking” are listed in the following Table 2, adapting her more detailed representation for the purpose but still maintaining much of the text as is as. (Langley, 1999). These are the exact strategies that can be used to carve the process out from the activities and events.

Table 2. *"Seven Strategies for Sensemaking" (simplified for purpose and adapted from Langley, 1999).*

Strategy	Key Anchor Point(s)	Fit with Process Data Complexity	Specific Data Needs
Narrative	Time	Fit with ambiguous boundaries, variable temporal embeddedness, and eclecticism.	One or few rich cases. Can be helped by comparison.
Quantification	Events, outcomes	Focus on events and their characteristics. Simplifies ambiguity away.	Needs many similar events for statistical analysis: one or few dense cases is best.
Alternate Templates	Theories	Adaptable to various kinds of complexity.	Once case is enough. Degrees of freedom come from multiple templates.
Grounded Theory	Incidents, categories	Adapts well to eclectic data and ambiguity. May miss broad high-level patterns.	Needs detail on many similar incidences. Could be different processes or individual-level analysis of one case.
Visual Mapping	Events, orderings	Deals well with time, relationships, etc. Less good for emotions and interpretations.	Needs several cases in moderate level of detail to begin generating patterns (5-10 or more).
Temporal Bracketing	Phases	Can deal with eclectic data but needs clear breakpoints to define phases.	One or two detailed cases is sufficient if processes have several phases used for replication.
Synthetic	Processes	Needs clear process boundaries to create measures. Compresses events into typical sequences.	Needs enough cases (5+) to generate convincing relationships. Moderate level of detail needed for internal validity.

Few concepts in the Table 2 deserve more explanation, even though intention is not to summarize entire article by Langley. Before presenting the strategies for "sensemaking", Langley discusses the nature of process data gathered in real operating environments. She states that "Process data collected in real organizational contexts have several characteristics that make them difficult to analyze and manipulate." (Langley, 1999). Following this idea, strategies listed by Langley have some criteria for the data, kind of "gate" for a particular strategy to be applied, or not. Some of the listed strategies work well for instance already with one case and with ambiguous data available from the process, whereas others may require a lot of similar event samples to yield reasonable results (e.g., to be able to use statistical analysis, or to be able to recognize patterns). The practical implication is that not all the listed strategies work equally good for all the processes there could be.

Applying a strategy should hence be decided based on what can be observed about the available process data from real operating environment. Categories of the process data complexity are opened more in the next sub-chapter, but “Fit with Process Data Complexity” column in previous Table 2 tells the expected fitness of each strategy to form a generic process, based on the nature of data under analysis. (Langley, 1999).

Last column of the table, “Specific Data Needs”, lists more specific details on data, e.g., how many cases are needed. Regarding this study, there is of course one major case for solving the research question but on the other hand the repeating cycles of DSR process can very well be considered separate cases: each iteration is a case of its own, and has different set of events, actions taken, and selections made.

3.3.2 Process Data Complexity

To be able to select a proper strategy to expand the understanding of DSR iterations, we must understand the criteria as defined by Langley. When looking at the process data complexity Langley lists several factors: first, data consists of events: she claims that analyzing process data events requires going beyond mere variance theories where different variables impact the phenomena. With process, she says, “…process theories provide explanations in terms of the sequence of events leading to an outcome (e.g., do A and then B to get C)”, compared to variance theories where “…more of X and more of Y produce more of Z” (Langley, 1999). Langley states that “The analysis of process data, therefore, requires a means of conceptualizing events and of detecting patterns among them.” (Langley, 1999).

Second form of complexity is that data has “…multiple levels and units of analysis whose boundaries are ambiguous”. (Langley, 1999). For instance, what belongs to some particular process and what does not? Some available strategies can deal better with this kind of ambiguity.

Third, Langley talks about “variable temporal embeddedness” of process data. (Langley, 1999). What this means is that there are a lot of factors that are not getting recorded as incidents even though they have underlying influence on the events being observed. Even very different level of events can have an impact here. Langley opens this by giving examples: “…an event may include a bad year, a merger, a decision, a meeting, a conversation, or a handshake.” (Langley, 1999). These are very different level of events, some maybe not recorded, but still having an influence on what we can observe.

Last category of complexity is the “eclectic” nature of data: this means the very diverse nature of process data: it can contain variables with events data, as well as “…the evolution of relationships between people or with the cognitions and emotions of individuals as they interpret and react to events” (Isabella, 1990; Peterson, 1998, according to Langley, 1999). (Langley, 1999).

When referring to the Table 2, the existence or absence of these factors in data can determine how well some strategy can be used when trying to define the process out of the data. Or they may lose or simplify certain aspect of the process if applied to certain type of data. Overall, strategy should be picked taking the nature of available data into consideration.

3.3.3 Selecting Alternative Strategies

Considering the DSR process cycles that have taken place with this thesis, all the levels of complexity are present, at least on some level. Regarding the data ambiguity, we are combining various disciplines in iterative manner and trying things out – this is ambiguous. Event leads to another: for instance, making certain technology selection precedes the following steps, instead of set of standard variables being always present. “Variable temporal embeddedness” has clearly been present in repeated DSR cycles also: certain changes in the underlying platforms have mandated a specific path of actions to be taken, and this was bound to actual timing of doing the technological experiments. Another time, it would have been different story and path. Last category of “eclectic” data applies on some level, relationships between people are at play as new information to research community is generated. Cognitive processes part of Design Science Research process are also described by Vaishnavi and Kuechler (2015, p. 17).

Regarding the number of available cases (one factor to consider when selecting a “sensemaking” strategy), it was already claimed before that it appears natural to consider the DSR process to contain “multiple cases” due to repeated, but different process cycles which each have different events and choices. Due to human choices made as a part of process the iterations are not similar with each other.

Deriving from the above discussion, the strategies that seem applicable to perceived DSR cycles are listed in the following:

- **Narrative strategy** and **Alternate templates strategy**: deals well with all levels of complexity present in DSR process. Also, data needs get fulfilled.

- **Ground theory strategy**: deals well with the data complexity. Might be challenging due to requirement of detailed similar incidences (for codification and categorization).
- **Visual mapping strategy:** deals relatively well with data complexity and allows abstractions. Weakness on emotions is not a factor with DSR, whereas the same on interpretations could be, depending on the case.
- **Temporal bracketing strategy**: could work as eclectic data can be dealt with. Need to have clear temporal breakpoints (in process data) may be bit more challenging. On the other hand, those are already available in defined DSR process.

Following strategies are not considered:

- **Quantification strategy**: requires many similar events to enable statistical analysis, and “eschews ambiguity”. (Langley, 1999). This does not look like good fit for DSR.
- **Synthetic strategy**: yields into definitions that are more “variance theories” than process models. We don’t want to focus on this now.

Out of the available options we choose to apply Narrative strategy. Idea of narrative strategy is to create “a detailed story from the raw data”, and in best case come up with ideas that could be applicable in other situations also. (Langley, 1999). Langley states that “In the hands of an accomplished writer, this sensemaking strategy has the great advantage of reproducing in all its subtlety the ambiguity that exists in the situations observed.” (Langley, 1999).

3.3.4 Observations on Design Science Research Process Based on Narrative Strategy

Even though the results of the thesis will be described in later chapter, we already here open some of the defining moments during the process, on relatively generic level. We limit the details for the sake of brevity and to avoid the repetition in later parts.

Whenever certain observation has significance for a particular result, listed in chapter “Summary of Results”, it is marked like this: ›› **DSR_RESULT99**, meaning this part of the text would contribute into (DSR-specific) result identified with DSR_RESULT99.

We use the narrative strategy to see if we can find additional insights to DSR process cycles, by observing through narrative storytelling what happened and how things were perceived. For that reason, the remaining of the chapter has more personal tone than other chapters in this thesis and uses the first-person pronoun in description.

Note! It is important to notice that the observations in this chapter don't concern the technical topic of the thesis or research question, but the DSR process itself. That is also why observations are categorized according to DSR steps. Also, even though approach was closer to earlier-mentioned DSR-IS, we reflect against the DSR process cycle, to be able to analyze initial steps like "Awareness of Problem" and "Suggestion".

Step: Awareness of Problem

As the selection of the subject for thesis became relevant, generative AI had just become hot topic. It seemed that majority of my LinkedIn-feed dealt with the AI, many of the discussion topics mentioning ChatGPT specifically. Also, it seemed that suddenly a vast number of AI experts had emerged from somewhere as so many people had AI mentioned in their slogan in a way or another. That is one rapid change of the tide! Having had the idea to do my thesis "about AI" already from the very beginning of my studies made it anyway easy to jump into the wagon.

As narrowing down the subject for the thesis with my professor, the first idea ended up being anything but narrow, rather it was very ambitious: "How can company data be provided to AI so that it can become extra board member?". The thinking behind was that generative AI had proven to be approachable interface to AI, and there's always more data than we can build analytics for via specific effort. However, it proved challenging to find business sponsor (i.e., data provider) for this idea. After very short consideration the option to use only synthetic data was abandoned, as that would not adequately simulate the real complexity. Later, more specific problem – and this time more narrow - with commercial interest behind it was presented. That became the topic of thesis.

Observation1: Interesting problems can be plenty, but missing sponsorship could prevent pursuing for the solution if ingredients needed for building the solution (like data for instance) are now readily available.

Observation2: There can be iterations almost on each phase of the process, for instance repeated process-internal dialogue between the step "Awareness of problem" and research effort proposal.

Neither of the above two observations bring new insights to the DSR process, however. First one is "fact of life" type of situation, and another one something I could expect with creative process.

Step: Suggestion

Suggestion step, following the Awareness of problem, should produce tentative design. This design does not need to include implementation but at least somewhat matured idea how we want to go for solving the problem. Vaishnavi and Kuechler mention that ”Suggestion is essentially a creative step wherein new functionality is envisioned based on a novel configuration of either existing or new and existing elements.” (Vaishnavi and Kuechler, 2015, p. 15).

Already before the actual research question had got narrowed from “let’s make AI our board member” to something a bit more limited, I had pondered alternative design options regarding the training of AI / LLM with company data. First thought was to train AI model with the data in local environment at the university, but that option was abandoned due to couple of main reasons: while one dedicated physical server would have been available, security was seen significant risk as the expectation was that potentially sensitive company data would be handled. Then, time and effort to train the model with the specific data was expected to be very high. In parallel, I sent a request for access credentials to one company that had just published a new LLM which seemed a potential option to the task. The response never arrived, though. Overall, the idea to use local resources was discarded and focus moved towards cloud and pre-trained models.

Next, I studied available offerings from one of the major cloud providers. Their technology would have been a good match with the company I was at that time discussing with, but this deal was never closed. Another challenge was that models available by the cloud provider had limited language support. It seemed important to have a native language support as for instance sentiment analysis was already at that time one of the expected functionalities, even in Finnish. There is always a possibility to translate the text automatically, but I thought it would be easy to lose nuances in the process. That, in turn, could be essential for detecting sentiments.

While discussing with the representative of the mentioned potential sponsor company, I was made aware of one design pattern which sounded extremely interesting as it was able to combine the rich conversational capabilities of the ChatGPT and still base its responses on the company data only. Provided reference design included also easily approachable web interface. Once the research question got clarified and discussions with another company settled, I was able to use this prior investigation and propose this as a starting point for the tentative design very quickly.

It was well known by that time that there are limitations in the size of input data that LLM can handle, even though the limits are growing with newer models. It was also known that one of the design patterns for handling these kinds of situations is to handle input data in pieces. This was not part of the reference design but something I expected to build myself. I was able to find another reference implementation that was called recursive, but it rather looked a looped implementation producing shorter summaries from pieces of large document and saving those shorter pieces into document.

Observation3: Tentative design step was going by the book in here, meaning that the target design was "...envisioned based on a novel configuration of … new and existing elements." (Vaishnavi and Kuechler, 2015, p. 15).

Step: Development

Development "step" looked rather a phase, not a straightforward step to an artifact. Of course, DSR process model does not claim anything about the duration or amount of work, either. Still, it seemed that as both solution domain and problem domain maturity were relatively low, a lot of things kept on moving and changing rapidly. Existing artifacts (mostly the reference design in this case) kept on changing almost constantly (even though I only merged codebases few times when necessary).

On few occasions earlier configurations stopped working, and rework had to be done to get back on track. For instance, geographical regions where ChatGPT model or one other module were available, changed. On one case, implementation that had been working previous night didn't work the next day when re-deploying to cloud. This was due to dependency module changes, and pulling the latest updates for reference design changes and bug fixes was required to continue, calling for transfer of own implementation on top of new codebase again. Not everything was easily mergeable as initial functions were also modified while looking for solution. What this all meant was that there were considerable number of iterations within the Development step itself, even before reaching the state of suggested by tentative design.

Observation4: Environment is not (usually) fully under one's own control. In case of using existing artifacts (e.g., reference design) path to artifacts, implementing tentative design, can be less straightforward due to changing dependencies. **›› DSR_RESULT1**

Observation5: When maturity of solution and problems domains are yet low, the concept of "existing artifact" may be evasive. New things get introduced constantly, and today's innovation is that no more tomorrow. **›› DSR_RESULT2**

Observation6: For someone who has mostly been working in the projects applying agile methodologies, the single cycle of DSR process seems very much waterfall (but this cycle is getting repeated, though).

Observation7: The expectation in DSR is that after each Development step, formal Evaluation and Conclusion activities are taken before the next cycle. I would claim that sometimes the evaluation and conclusions could be informal, sudden, even binary: “this is not good... this does not work!”. In my opinion this could in certain cases lead to instant reworking of the (design and) implementation without going through the formal steps of Evaluation and Conclusion, if clearly no relevant design science knowledge would be generated. This is certainly what I did myself. **›› DSR_RESULT3**

Observation8: DSR allows working with the design as well as with the implementation within the Development step. But why not to show it in the process model itself, as to me this sounds iterating within a step?

Following the Observations 6, 7 and 8 I searched if something has already been stated on DSR and Agile. In their conference paper Conboy *et al.* (2015) propose Agile Design Science Research Model, ADSRM. This work is proposing to bring elements from Agile into DSR in wider scale, but still proposes iterating each cycle fully. On the other hand, DSR-IS as suggested by Vaishnavi and Kuechler (2015, p. 60) seems to be less strict on the exact steps, rather focusing on included activities.

In my mind DSR allows the “hidden” iteration within the Development step as both design and implementation can be matured. Whether to make this visible or not has two angles: making it visible might give too much permission to skip the Evaluation and Conclusion steps sometimes when there really would be benefits in taking those. Making it visible (and perhaps optional) would agree that sometimes there is no new design science knowledge available after dead end. **›› DSR_RESULT4**

Steps: Evaluation & Conclusion

My way of working admittedly was not pure DSR, and not ADSRM either. Perhaps mostly the DSR-IS. In pure DSR the requirements for final two steps in a cycle are quite comprehensive: things are carefully recorded, explained, and analyzed. However, I feel it is less evident how to carve important design science knowledge out of the dead end where the conclusion simply is that “this is not working!”, if the reason is not some technical curiosity but for instance poor idea how to implement something, or the change of the platform underneath that has rendered previous version as non-working, despite all the configuration settings in place. I can, however, identify few grand DSR cycles during the technical part of the thesis (while great number of iterations within the Development step, like explained in previous chapter), so in macro-scale the process used is DSR.

What seemed to be at least an improvement somewhere in the middle of the journey was not necessary anything like that as time passed. Even the end state of the research gets challenged via constant improvements: for instance, during the making of thesis the capability to analyze PDFs via ChatGPT, or with Bing (using ChatGPT in the background) as a part of Microsoft Edge browser was introduced. Today's innovation can be business as usual tomorrow, and old news soon after.

However, I feel that principles gained during the process can remain valid for much longer as they can be utilized also in future. This is well described by the earlier Figure 8 about design science knowledge hierarchies, even though there the emphasis is really on the design science knowledge produced for the community and I am more talking about personal competences.

In the end of the Design Science Research process, we may have outcomes that can be considered as repeatable "facts", or "loose ends" that cannot be so well explained. (Vaishnavi and Kuechler, 2015, p. 17). I certainly have loose ends as well, for instance related to prompt engineering. Why do certain things seem so much harder to "teach" to LLM than others, no matter how guided? Why do results sometimes vary more than expected when nothing was really changed? While I think I have full control of the behavior of things visible to me, do I really? For instance, is the version of the LLM I used ("gpt-35-turbo") behind the API technically same today and tomorrow? Have implementation of other building blocks in the cloud changed and is it their mutual play that hit me?

Finally, I found out that some of the loose ends may start to turn into less loose, even towards the very end of the documentation process. This happened to me with some difficulties faced on prompt engineering, especially related to counting things, like amount of given feedback. While copy-pasting the text from the generated PDF (Portable Document Format) file, I found out that some of the whitespace-characters (line breaks etc.) were off, even though on surface PDF seemed fully ok. This can explain the difficulties with counts at least to some level. Research is never ready!

Observation9: In a longer project the outcome of the intermediate Conclusion step that seemed to produce new insights, may change before reaching the end of the project. **›› DSR_RESULT2**

Observation10: Design science knowledge that leans towards the technological end of the spectrum can have its "best-before date" coming soon, especially in domain that has low solution and problem maturity. Principles, however, will last longer. This, however, was already stated in the literature. ›› **DSR_RESULT5**

Observation11: Loose ends do seem to get created. Feasible explanations (that could be matured into repeatable solution) do not always present itself. Mechanisms for those could vary depending on the project of course and are many.

Observation12: Loose ends could turn into less loose even in the very end of the process. Sometimes explaining and documenting the results may open new insights. A little bit like with GPT – they yield better results when they think through step-by-step what they are supposed to do. ›› **DSR_RESULT6**

Observation13: The existence of more relaxed forms of DSR, like DSR-IS, seems to highlight the fact that one size does not fit all. There are scenarios where tuned approach works better.

4. SYSTEM DESCRIPTION AND DESIGN PHASES

The concrete, tangible target of the thesis (POC software), as well as technology platform selection process (around topics like whether training or finetuning own AI etc.) have already been opened earlier, so they will not be repeated anymore in detail. Instead, this chapter will describe the high-level starting point architecture, essential parts of the system used and built, and some key methods and/or functions. The work will be presented through few logical chronological phases, or DSR process iterations. Most of the code extracts are given for the last version of the POC, but some also earlier phases.

Some code presented in this, and several following chapters, may have been formatted for the page for clarity and readability. Also, some draft comments may have been edited for clarity, but this does not change the functionality. These practices apply to this section and all the following sections with code samples, except code in Appendix D which is almost unedited.

Whenever certain paragraph or comment has significance for a particular result, listed in chapter “Summary of Results”, it is marked like this: ›› **RESULT99**, meaning this part of the text would contribute into result identified with RESULT99.

While doing the thesis, the search paradigm was under heavy change. Let’s face it, Google has been a trusted friend of every software developer. Now, instead of giving mere keywords to Google to find some aid for coding related issues, the conversation with Microsoft Bing (now called Copilot, integral part of Edge browser) became possible. For the sake of transparency, it needs to be mentioned that these new ways of working were indeed utilized when doing this thesis. It is also part of the learning itself. Suggested code snippets were used for thesis and related POC software. Also, AI was great assistance explaining the themes and ideas of code snippets of existing code when familiarizing with it. Further, at least on one occasion it even helped with specifics when maturing the idea, but this of course required asking right questions and already having the idea in mind where to go. These changes in coding paradigm and even in generic way of working are widely discussed in public at the moment and expected to change for instance the nature of software development. However, responsibility is never transferrable to any AI-assisted code-pilot, checking the outcome is always the responsibility of individual. Same is true here.

Also, other assistance was used while developing the software and documenting it, for instance dictionaries, Microsoft Word spell-checker and thesaurus and so forth.

System was developed in three main logical phases which are opened in following sub-chapters. This does not mean that there would be three commits in repository, far from that. Rather, they are logical steps that can be seen retrospectively. Within these steps there were some micro-iterations and even change of mind. Still the goal is to represent the essential outcomes and decisions here.

4.1 Phase 1

First phase was about getting the basics to work. This includes being able to deploy the application to the cloud environment and running it successfully, as well as getting started with summary generation, even with documents of lesser length.

Several things introduced in Phase 1 did not live through the entire lifecycle. This, however, is acceptable due to iterative nature of DSR process.

4.1.1 Reference Platform Software and RAG-pattern

Microsoft's reference architecture available via GitHub was used as a starting point ("ChatGPT + Enterprise data with Azure OpenAI and Cognitive Search," 2023). Software is available under MIT License which can be found in Appendix A of this document. The version of the software available around the end of July 2023 was used as a starting point for the Proof-of-Concept software. Figure 10 is taken directly from the repository and represents the high-level architecture:

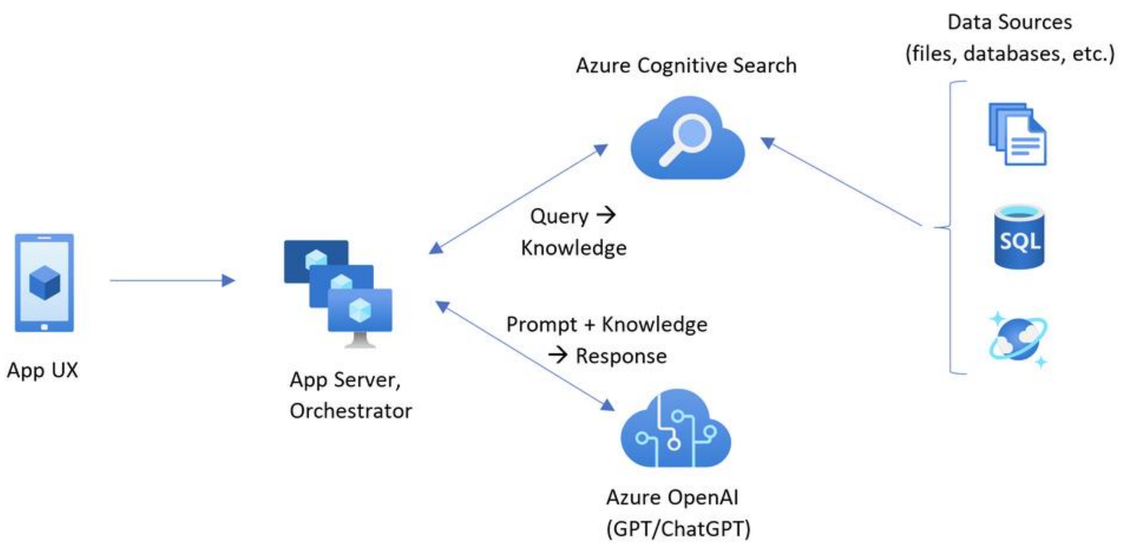


Figure 10. *High-level system architecture ("ChatGPT + Enterprise data with Azure OpenAI and Cognitive Search," 2023).*

The design pattern used here is called as “Retrieval Augmented Generation”, which is explained and demonstrated in video related to the topic (*Making Enterprise GPT Real with Azure Cognitive Search and Azure OpenAI Service*, 2023). Further background is given in a dedicated blog post from March 2023, even though the software has since gone through significant evolution (Castro, 2023). “Retrieval” in the name of the design pattern refers to the part where section(s) of indexed document(s) – represented as “Data Sources” in previous picture - are retrieved from the search service (Azure Cognitive Search), and this data is then used to Augment the answer using the Azure OpenAI model (ChatGPT) via API. Augmentation happens in the backend and output is provided to user via web-based user interface. What is fundamental in this design pattern is that it does not require training of AI-model with company data, any pre-processing, organizing or translation of source data, but model can be used as is. (Castro, 2023; *Making Enterprise GPT Real with Azure Cognitive Search and Azure OpenAI Service*, 2023). One point to add is that in later phases the Retrieval part was changed away from Search Service.

Sample interaction, using the software as-is, is provided in the GitHub repository readme file, shown in next Figure 11:

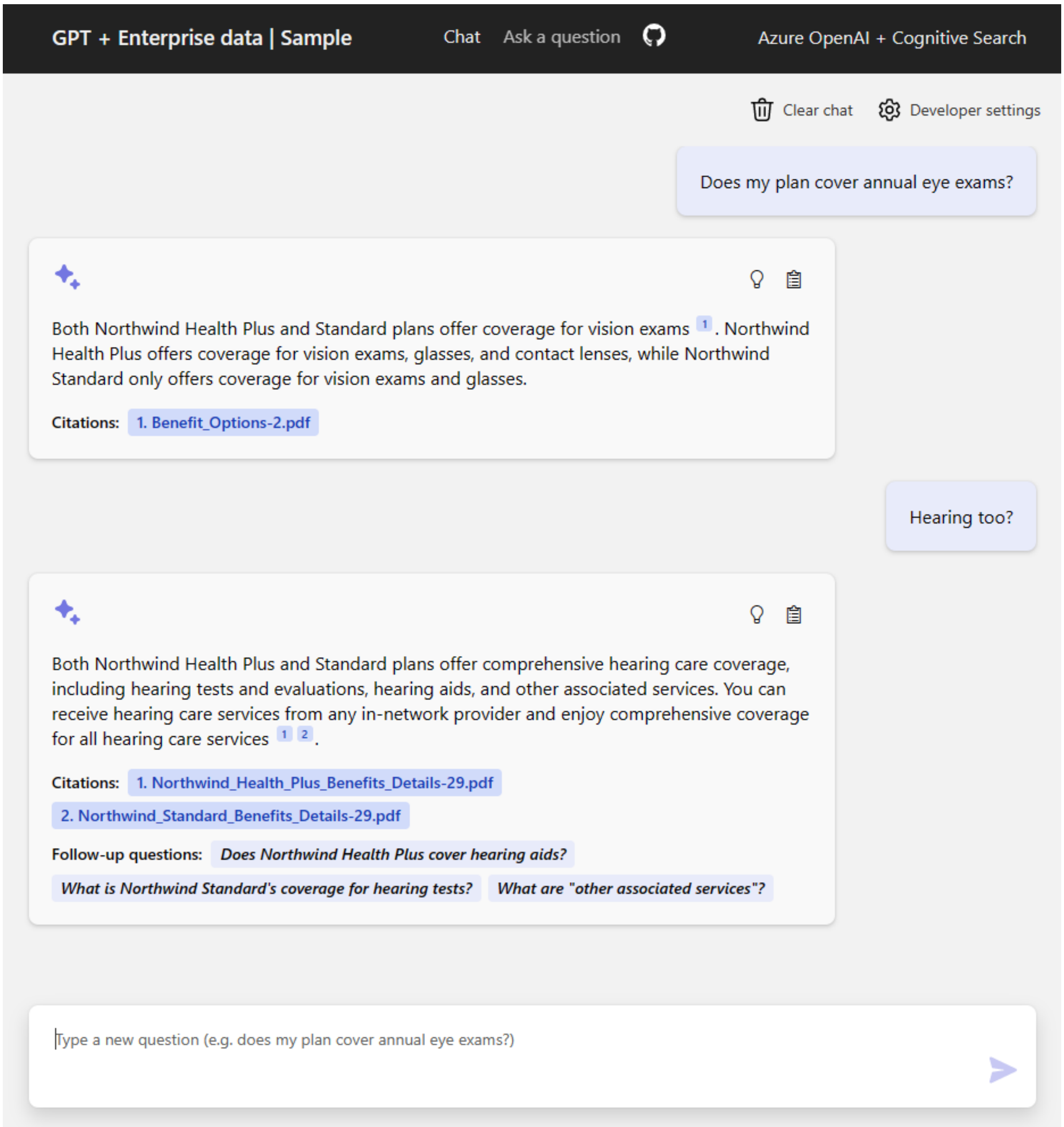


Figure 11. *Screenshot of the sample run ("ChatGPT + Enterprise data with Azure OpenAI and Cognitive Search," 2023).*

Software included few alternative sample "interaction patterns" or "approaches", meaning different ways to design prompts, build queries, and interact with the model. Examples were simple "retrieve-then-read", which is said to work well for simple Q&A scenarios as question has all the information needed to fetch matches from search index. In practice "retrieve-then-read" means that "…approach simply uses the question to retrieve from the index, take the top few candidates, and inline them in a prompt along with instructions and the question itself." (Castro, 2023). Creation of POC software was started on top of "ChatReadRetrieveReadApproach" (of type "read-retrieve-read") as that was utilized in chat functionality. It is multi-staged approach that uses ChatGPT to build a query based on earlier discussion history, fetches relevant documents from search service based on this query and then provides the original question with conversation history and fetched data to OpenAI API to get the response. Example of this can be seen in the previous picture, where query “Hearing too?” would not make any sense if earlier context was not considered. (“ChatGPT + Enterprise data with Azure OpenAI and Cognitive Search,” 2023; Castro, 2023).

The concrete boundary for how much history can be carried around is set by the token limit of the used model, “gpt-35-turbo”, is 4096 tokens (“Azure OpenAI Service models - Azure OpenAI,” 2023). This set of 4096 tokens must include all the history, system prompt (behavioral guidance), user’s last prompt (e.g., a question) and even completion of the prompt (e.g., answer to the question) itself.

4.1.2 Data

Original source data (from the company that this thesis is made for) is by human and varies heavily in quality. When studying the data, part of it did not make any sense, literally, but was only garbage, being apparently random characters after each other. Company representative opened the background explaining that in certain cases data created by mandatory surveys was of poor quality, instead of indicating any technical issue. In addition, there were also entries repeated multiple times, some entries were longer, but off-topic, and so on. There also were relevant replies for the actual questions. For confidentiality reason actual customer data will not be represented in here.

Artificial, synthetic data (data that was in fact generated by ChatGPT) was decided to be used to proceed with the study. This enabled to focus on solving the questions at hand instead of using time on cybersecurity measures when deploying test software and executing it. Uploading actual customer data to the cloud without implementing necessary security measures was seen too risky. This prevented the possibility for real source data leaking out. English version of generated data used in Phase 1 is shown in Appendix B. Similar type of Finnish version of generated data was used.

4.1.3 Extracting Insights and Detecting Sentiments

Already during Phase 1 test drives were made to understand how the basics for insights and sentiment detection work. Had it seemed evident that the selected approach is not working towards intended targets, different direction would have been required. Figure 12 shows example run of the system, using the version of generated data shown in Appendix B:

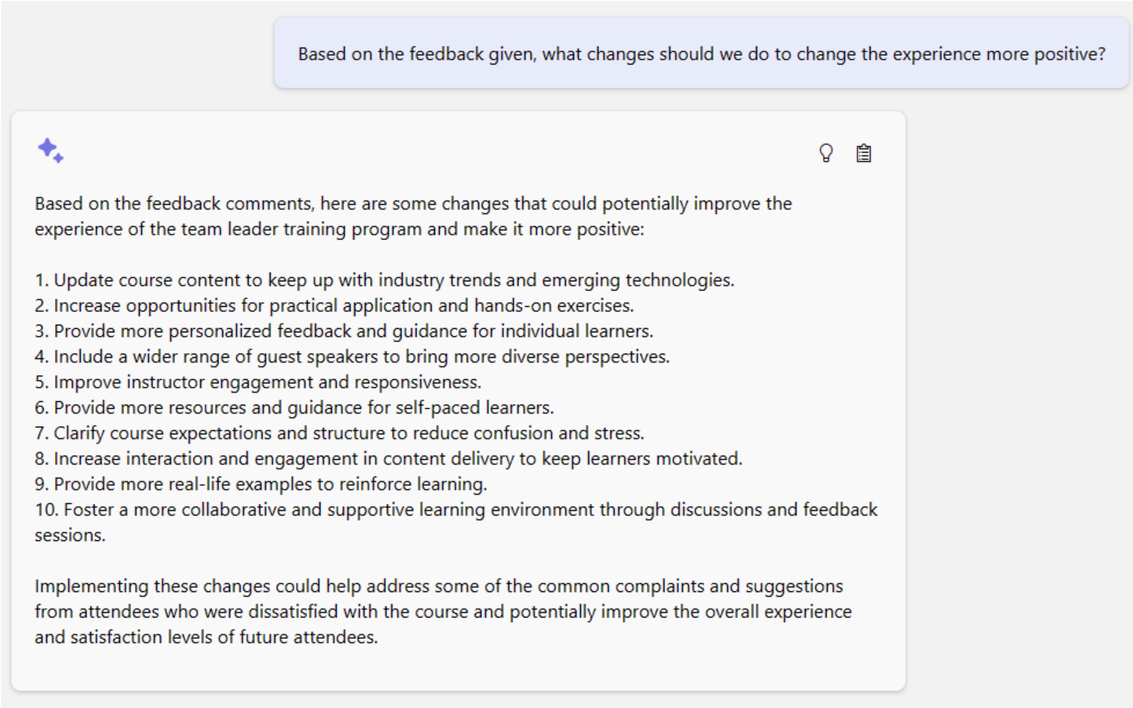


Figure 12. *Phase 1 sample run for extracting insights from data.*

Looking at the user question made, and comparing the given recommendations to the data source, the recommendations seem to make sense and match the content well. See the comparison in the next Table 3:

Table 3. Observed correlation between data and AI output.

Sample Extracts from Data	Recommendations Generated by AI
"…the content felt outdated.", "The content was outdated…", "The lack of up-to-date industry examples made it difficult to relate…", "The course curriculum was outdated, failing to keep up with industry trends and emerging technologies."	Update course content to keep up with industry trends and emerging technologies.
"The course … lacked practical application. More real-life examples would have been beneficial", "The course failed to provide sufficient practical application opportunities. More hands-on exercises would have reinforced the learning."	Increase opportunities for practical application and hands-on exercises.
"The course failed to provide sufficient opportunities for individualized feedback. Personalized guidance would have enhanced the learning experience."	Provide more personalized feedback and guidance for individual learners.
"Including a wider range of guest speakers would have enriched the learning experience."	Include a wider range of guest speakers to bring more diverse perspectives.
"…the support was lacking.", "…there was limited interaction.", "…feedback was vague.", "…instructors were unresponsive…", "The course fell short in terms of instructor engagement. More active involvement and timely responses would have improved the experience."	Improve instructor engagement and responsiveness.
"The course lacked support for self-paced learners. More resources and guidance for independent study would have been helpful."	Provide more resources and guidance for self-paced learners.
"The course structure was confusing, and the expectations were not clearly communicated. This led to unnecessary stress and confusion."	Clarify course expectations and structure to reduce confusion and stress.
"…there was limited interaction.", "The course lacked interactive elements, making it challenging to stay engaged throughout the duration."	Increase interaction and engagement in content delivery to keep learners motivated
"The course … lacked practical application. More real-life examples would have been beneficial.", "… lack of up-to-date industry examples made it difficult to relate the content to real-world scenarios."	Provide more real-life examples to reinforce learning.
"…there was limited interaction.", "… course failed to provide sufficient opportunities for individualized feedback."	Foster a more collaborative and supportive learning environment through discussions and feedback sessions.

As can be seen in Table 3, data source and recommendations correlate well, even if the system prompt (AI behavioral guidance) at this time was not on the level of finished Phase 1 system prompt. RAG-pattern was at use and relevant responses are given based on data injected part of the prompt. ›› **RESULT1.**

One thing to be said, though, is that other comments in data (Appendix B) did praise the same things that other comments criticized. For instance, the recommendation was made for increasing *"…practical application and hands-on exercises."* However, there were several comments in data that were favorable for the same subject, for instance the following: "*The course offered a good balance of theoretical concepts and practical application. The hands-on projects were particularly helpful.*". This AI did not comment in any way, but it was not asked to do that, either. Same applies to comment about adding guest speakers, and many others. Feedback material is of course synthesized and may not reflect real scenario. In real scenario feedback could be more aligned, but experiences are always personal, so this is not guaranteed.

Even before previous test (meaning using the different version of system prompt), it could be demonstrated that multiple languages can be supported out-of-the-box due to the capabilities brought by ChatGPT. Figure 13 shows the example discussion where the software is replying in English and in Finnish:

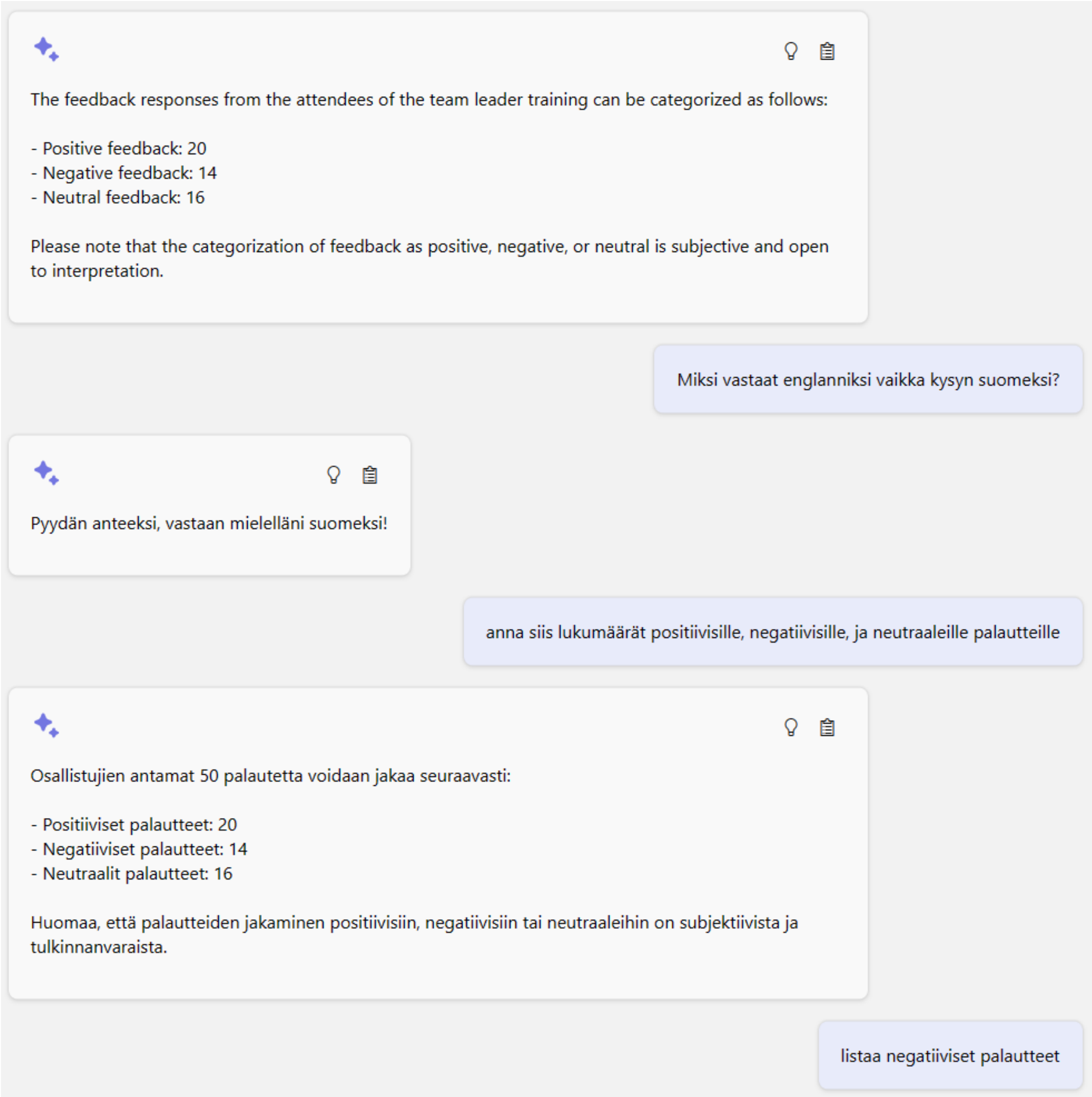


Figure 13. *Phase 1 sample run for using multiple languages.*

This use case itself is not of showroom quality yet, for instance AI replied in English even when asked in Finnish. There are few possible reasons for that at this early stage. However, it still already demonstrates that also Finnish can be used in conversations. ›› **RESULT1**

The last request in Figure 13 in Finnish translates as: “please list the negative feedback”. The outcome of that is presented as a continuation in next Figure 14:

Tässä ovat negatiiviset palautteet, jotka on listattu lähdemateriaalissa:

3. Disappointing experience. The course lacked depth, and the content felt outdated.

5. Mediocre at best. The course material was too superficial, and the support was lacking.

9. Regret wasting my time and money on this course. The content was outdated, and the instructors lacked expertise.

12. This course fell short of my expectations. The assignments were repetitive, and the feedback was vague.

16. I had high hopes for this course, but it failed to deliver. The lack of organization and coherence made it difficult to follow.

19. This course was a waste of time. The instructors were unresponsive, and the course structure was chaotic.

29. The course failed to meet my expectations. The lack of up-to-date industry examples made it difficult to relate the content to real-world scenarios.

34. The course failed to provide sufficient practical application opportunities. More hands-on exercises would have reinforced the learning.

39. The course suffered from inconsistent communication. Clearer instructions and timely updates would have improved the overall experience.

42. The course curriculum was outdated, failing to keep up with industry trends and emerging technologies.

46. The course failed to provide sufficient opportunities for individualized feedback. Personalized guidance would have enhanced the learning experience.

49. The course structure was confusing, and the expectations were not clearly communicated. This led to unnecessary stress and confusion.

Figure 14. *AI listing the feedback considered to be negative.*

Few observations can be made from Figure 14. First, all the listed feedback really has negative tone in them, so fundamentals of sentiment detection work. For instance, totally positives are not mixed with negatives.

Like stated even by AI above in earlier Figure 13, categorization of items to sentiments is subjective, though. If we compare the listed feedback against the Appendix B which shows the data being used, we can (subjectively) say results appear to be relatively good. There are still few other feedback entries that could be listed as negative, as they include complaints. For instance, feedback 23: “*The course did not meet my expectations. The lack of structure and inconsistent pacing made it difficult to follow.*”. But someone else could of course see this as neutral observation, rather that emotional negative burst. AI did not list item 23 above.

Another observation is that AI listed 12 feedback, even though earlier it said (in Finnish) that there would be 14 negative feedback. Logs are not available to deeply investigate reasons, but one possible explanation is that as data gets read and repeatedly processed between queries the history is cut out, due to context window size limitations. This is more probable to happen when there is more data and lengthy system prompt. In practice this means that AI cannot take history into account and instead responds to prompt as a new thing, without much past knowledge. In such case part of the things it has said earlier, will simply drop off. Another observation is that in Phase 1 the “temperature” parameter of AI (impacting variations is responses) was still 0.7 which can give different answers to same question each time. Later this was dropped to 0.0 consistently.

Overall, there is no single right answer to categorize the feedback listed in Appendix B. For instance, how should one interpret the sentiment of feedback 14: “*The course had a strong theoretical foundation but lacked practical application. More real-life examples would have been beneficial.*”? This has elements of praise (“*…strong theoretical foundation…*”), negative expression (“*…but lacked practical application.*”), and even recommendation that could be seen as neutral (“*More real-life examples would have been beneficial.*”). Comparing the answers and source data, AI can be said to perform reasonably well when interpreting sentiments. ›› **RESULT1**

4.1.4 Fetching Entire Document vs Fragments

Reference implementation relied on search service returning the fragments of documents and using those as basis for completing the prompt. However, our need was to fetch the entire document. This was required because summary of the document must be based on entire content, not only a fragment, not even few of those. Any questions concerning the entire content of the file naturally needed the access to full text. To be able to do this, references to Azure blob client (client software operating with the file container), storage account and storage container were injected into ChatReadRetrieveReadApproach from another file, and logic was built to utilize those references. Next code snippet, Program 1, highlights the changes vs. the baseline software in the class constructor, where references were injected and stored into instance variables. Please notice that formatting has been adapted for page, and some draft comments removed!

```python
def __init__(
        self,
        search_client: SearchClient,
        blob_client: BlobServiceClient,
        chatgpt_deployment: str,
        chatgpt_model: str,
        embedding_deployment: str,
        sourcepage_field: str,
        content_field: str,
        storage_account: str,
        storage_container: str):
    self.search_client = search_client
    self.blob_client = blob_client
    self.chatgpt_deployment = chatgpt_deployment
    self.chatgpt_model = chatgpt_model
    self.embedding_deployment = embedding_deployment
    self.sourcepage_field = sourcepage_field
    self.content_field = content_field
    self.storage_account = storage_account
    self.storage_container = storage_container
    self.chatgpt_token_limit = get_token_limit(chatgpt_model)
```

Program 1. *Injecting required references into class constructor.*

This change enabled to fetch the entire file from the blob storage once match was returned by search service. Following method (Program 2) was added and it uses earlier injected references. The idea of the *get_document_from_blob()* method shown was to provide the iterator to the PDF-file so that it's content could be accessed page by page elsewhere within the class.

```python
# Async method for downloading the file from given blob to memory. Yields iterator.
async def get_document_from_blob(self, filename: str, storage_account: str,
storage_container: str):
    container_client = self.blob_client.get_container_client(storage_container)
    download_stream = await container_client.download_blob(filename)
    pdf_file = io.BytesIO(await download_stream.readall())
    pdf_reader = PdfReader(pdf_file)
    for page in pdf_reader.pages:
        yield page
```

Program 2. *Method for fetching the PDF from blob storage.*

Below code snippet, Program 3, demonstrates how above-mentioned method was utilized in a program to extract the full text from PDF:

```python
        filename = await filenames.__anext__() # NOTE, only first result returned
        pages = []
        async for page in self.get_document_from_blob(filename["sourcefile"], self.
        storage_account, self.storage_container):
            pages.append(page.extract_text())
```

Program 3. *Fetching the contents of PDF-file and storing into list.*

Code above takes the first filename returned by search service and then reads the content of that file using *get_document_from_blob()*, page by page into “pages” list. The logic of returning the filename(s) is not shown, but it much re-used the existing code, only selecting and returning the filename.

4.1.5 Generating Page Summaries

Relatively crude version for recursively handling the parts of the text was added. At this stage, the text of PDF-file was accessed page by page as can be need from Program 4:

```
# Async method to iterate and summarize the contents of the pages until the combined size
# is short enough to be fed into ChatCompletion for final summary.
async def generate_page_summaries(self, pages: List[str], last_user_prompt: str, history:
list[dict[str, str]], temperature: float):
    system_message = self.system_message_chat_conversation.format(injected_prompt="",
    follow_up_questions_prompt="")

    for i in range(len(pages)):
        page = pages[i]

        messages = self.get_messages_from_history(
            system_message,
            self.chatgpt_model,
            history,
            last_user_prompt + "\n\nSources:\n" + page,
            max_tokens=self.chatgpt_token_limit)

        chat_completion = await openai.ChatCompletion.acreate(
            deployment_id=self.chatgpt_deployment,
            model=self.chatgpt_model,
            messages=messages,
            temperature=temperature,
            max_tokens=1024,
            n=1)

        chat_content = chat_completion.choices[0].message.content

        pages[i] = "\nPARTIAL ANSWER:\n" + chat_content +"\n\n"
```

Program 4. *Recursive Phase 1 method for generating page summaries.*

Method *generate_page_summaries()* got the pages list as a parameter, and that list was iterated page by page. Chat history, page data (as injected data) and some other parameter was first stored into *messages* variable. Then, prompt completion was requested using the OpenAI API, and was result stored back into list, simultaneously indicating it was a partial answer only. This got repeated for each page and it is easy to see this way of working if very much sub-optimal, but it acted as a step forward.

Method described above was called elsewhere in code in simple while-loop in a way demonstrated by Program 5:

```
num_of_characters = sum(len(item) for item in pages)
passed_once = False
while num_of_characters > NUM_OF_CHARACTERS or passed_once == False:
    await self.generate_page_summaries(pages, history[-1]["user"], history,
    temperature=overrides.get("temperature") or 0.0)
    num_of_characters = sum(len(item) for item in pages)
    passed_once = True
```

Program 5. *Summarizing and condensing text via method looping.*

Via looping the length of the generated summary (which is the sum of the length of the text in all “condensed” pages) got smaller. Once this sum got below the given trigger value (NUM_OF_CHARACTERS) loop exited. Following that the final summary from ChatGPT API was requested, but this is not shown here.

4.1.6 System Prompt

Next aspect that required considerable amount of thinking and effort was the system prompt. System prompt controls the behavior of ChatGPT and can be fundamental to the results achieved.

In the first phase several trials were made to control the behavior of the POC software and get better (reliable and repeatable) results by tuning the prompt. For instance, explicit guidelines for summarizing or handling the sentiments were given. In addition, several examples for counting the feedback were given. Version of the prompt can be seen in Appendix C.

4.1.7 Other Aspects for Phase 1

In addition, Phase 1 contained other changes and additions. Those are shortly listed here without code samples.

- Remote debugging was not possible, so workflow was partially relying on output on Azure Web App console output. To support this logging was enabled.
- Reference software used indexing which was language-specific and had support for English only. As the target was to be able to use the POC also with Finnish

documents, some trials adding support was Finnish also were made. This required changes in few places, for instance in python-script preparing the documents (e.g., indexing and uploading to cloud), and in backend implementation which called the search service and ChatGPT API. Despite requiring some amount of work code examples are not given here as this was later dropped.

- In Python-script that is used to prepare and upload the documents into cloud few changes were also made. For instance, a new function that uploaded file into blob without splitting it was added, and another new function that indexed file as a whole, instead of fragments, etc. These were also dropped in the last iteration. Program 6 presents a code example for modified upload function that was used at this point, though:

```python
# This function uploads the PDF into blobs without splitting it.
# Re-uses the logic from original upload_blobs()
def upload_blobs_dont_split(filename):
    blob_service = BlobServiceClient(account_url=f"https://{args.storageaccount}.blob.core.
    windows.net", credential=storage_creds)
    blob_container = blob_service.get_container_client(args.container)
    if not blob_container.exists():
        blob_container.create_container()

    # upload files as single blob
    blob_name = os.path.basename(filename)
    with open(filename,"rb") as data:
        blob_container.upload_blob(blob_name, data, overwrite=True)
```

Program 6. *Phase 1 support function for uploading the document into blob without splitting.*

Function re-uses most of its logic from original upload function in repository. Uploading of the document is based on filename, pointing to pre-defined folder (data folder). If blob container does not yet exist it gets created, after which entire data gets uploaded without splitting it into separate file objects (blobs).

4.1.8 Analysis for Phase 1

Few observations were evident after the first phase:

- Various extra dependencies (from the research question point of view), like use of Azure Cognitive Search and indexing, splitting the documents into blob storage and so on, added with the changing implementation of the baseline software made the progress slower than it would have been without those aspects. On the other hand, baseline software provided a good starting point, especially in terms of merging the capabilities of GPT AI and own data.

- Generating the summaries recursively worked, but clearly the algorithm at this point was far from optimal. Handling the data was based on individual pages, instead of configurable size of text. But it looked evident that AI was able to extract grounded insights based on injected data. This is the benefit of the selected architecture, having capabilities of ChatGPT readily available. ›› **RESULT1**
- The importance of system prompt tuning started to become evident. Equally evident was that GPT does not always do what expected. Retrospectively it can be noted that at least in one part of the software temperature parameter value was clearly above 0.0 (actually 0.7) which can cause variation into output. But there are other aspects as well: 15th August 2023 Microsoft had added the following comment into code: “Model does not handle lengthy system messages well. Moving sources to latest user conversation to solve follow up questions prompt.” (Balakrishnan, 2023). This means that instead of adding the data sources (own data) in the end of the system prompt, it was added in the end of the conversation history, to tackle some issues caused by lengthy system prompt.
- Language-specific indexing did not feel quite scalable idea. Solving was left for later.

4.2 Phase 2

Second phase aimed to improve the understanding of the context of a text and bring some added performance into solution. All the observations from Phase 1 were not tackled in this phase yet.

Like the first phase, Phase 2 also included few items that were not taken to the last and final Phase 3. It also tested an idea of ideas for increased context awareness but with less success at this stage. Those ideas we matured in last phase, however.

4.2.1 Context Awareness

As already hinted via *generate_page_summaries()* explanation during Phase 1, it started to look evident that there was a need to somehow let AI model know that it would have access either to partial data or full, even if already condensed (iterated), data. Even if the intention indeed was to process entire document (full data), that had to be done in pieces (partial data) due to the token limit as explained earlier. The need to indicate these two different scenarios was based on finding that when asking a question relevant to the entire document, but AI having access only to partial data – which didn't explicitly contain information about the context – it sometimes got confused and replied in unexpected way. Figure 15 has an example (underlining added retrospectively to the picture):

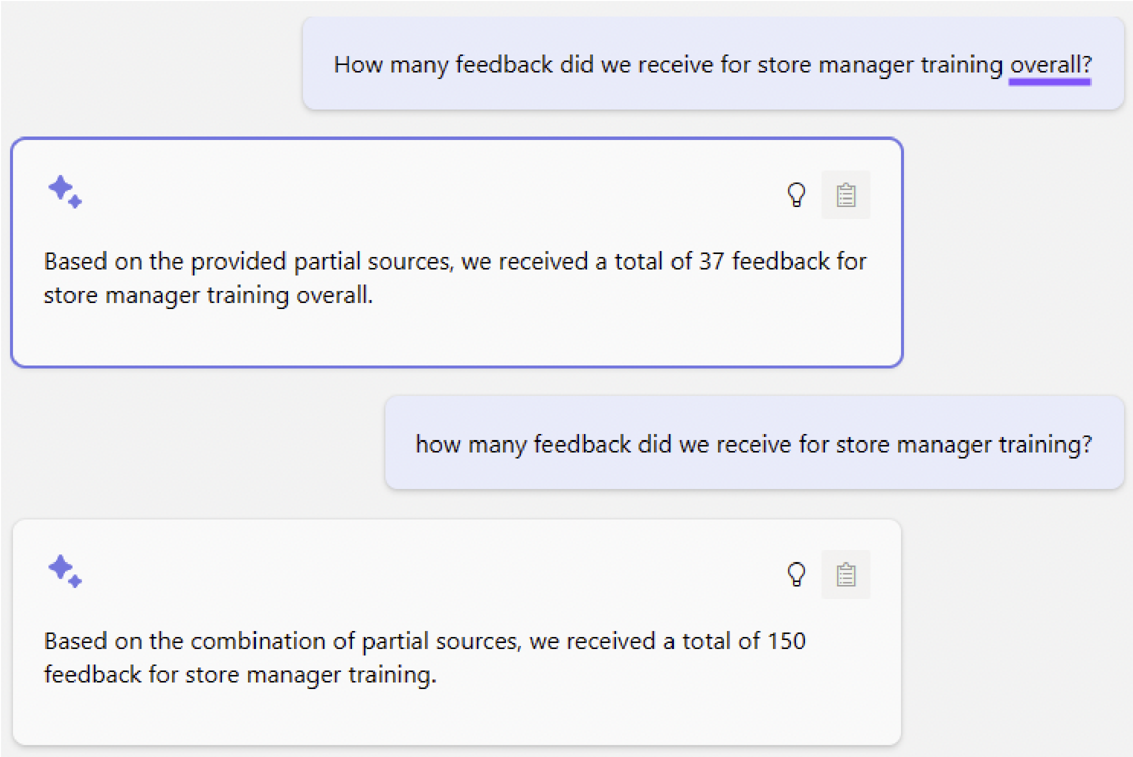


Figure 15. *Example of AI getting confused with term "overall".*

The system prompt used at this point was an evolved, but intermediate version of Phase 2, i.e., not the same as in Phase 1 / Appendix C. These two questions were made to the same document, and in almost the same way. The only difference is that in first question we have a word "**overall**" in addition. As can be seen the answers are very different, however. The latter answer, 150 feedback, is the right one for the version of data used at that time. This intermediate version of data is not shared in this document, as it is not relevant for this result, or the problem observed and explained next.

To understand what is going on we can peek into thinking process behind the scenes. Below Figure 16 shows what was happening in case of the first question-answer pair, which gave the incorrect answer of "…37 feedback…" (highlights added retrospectively to the picture):

```
{'role': 'user', 'content': 'How many feedback did we receive for store manager training
overall?\n\n Sources:COMBINATION OF PARTIAL SOURCES FOR FINAL ANSWER:\n PARTIAL SOURCE:
Based on the provided partial sources, we do not have information about the total number
of feedback received for store manager training. , PARTIAL SOURCE: Based on the provided
partial sources, we do not have the total number of feedback received for store manager
training. , PARTIAL SOURCE: Based on the provided partial sources, we received a total
of 36 feedback. , PARTIAL SOURCE: Based on the provided partial sources, we received a
total of 37 feedback for store manager training overall. , PARTIAL SOURCE: Based on the
provided partial sources, we received a total of 16 feedback for store manager training
overall. '}
```

Figure 16. *Confused thinking process when term "overall" was used in prompt.*

The snippet of the prompt shown in Figure 16 contains the question that user did input via UI ("How many feedback … overall?") – along with the collection of answers based on processing the parts of larger document. Each of these partial answers, labelled as "PARTIAL SOURCE", is the outcome of taking a slice of the document and processing it via GPT API. Following Figure 17 shows that part of the flow:

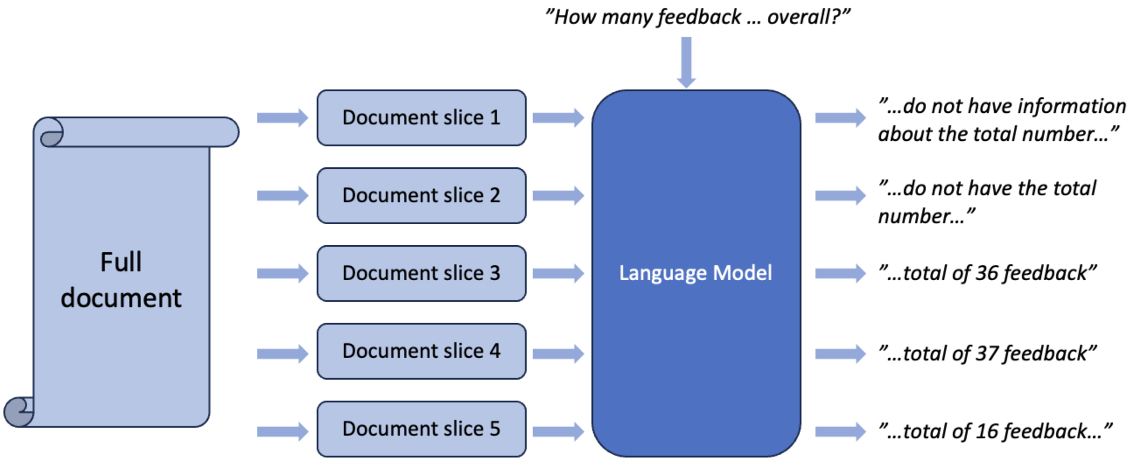


Figure 17. *Concept of handling large document in parts.*

This slicing is done simply due to the token limit. As depicted by the Figure 17 above, we aim to answer to the question separately for each slice of the document. What is not shown in the picture is that partials get combined to final answer.

After we have received answers for the parts, we combine them to final answer. In case of creating a textual summary, this would mean summary of summaries. In case of count of something, we should combine the answers by adding them together. To control this behavior, instructions in system prompt were used during Phase 2. As seen in prompt (see Figure 16, and term "COMBINATION OF PARTIALS SOURCES FOR FINAL ANSWER") there is a collection of parts available, to generate actual final answer to user's question. And as can be observed, AI has got confused in several cases.

We wanted AI to count the sub-sums together and give the final answer via UI. However, many of the partials were even missing the count altogether. When interpreting the thinking process via Figure 16, AI deduced that based on some document **slice only**, it simply cannot tell how many feedback we did get **overall** for the training. Hence then answer, “don’t know”. This is a little like asking children “How many ice-creams did you take overall?” and they might say “Not sure…” – because who can remember all the ice-creams ever eat in one’s lifetime? If asked how many **today**, though, the answer could have been four. Further, even though some parts included the amount of feedback in that partial, the intended adding up did not work either. Model only returned single (largest) of the individual counts given.

When then changing the question by only removing the single word, “overall”, the correct answer was given! Figure 18 gives a peek to the corresponding background process:

```
{'role': 'user', 'content': 'how many feedback did we receive for store manager
training?\n\n Sources:COMBINATION OF PARTIAL SOURCES FOR FINAL ANSWER:\n PARTIAL
SOURCE: Based on the provided partial sources, we received a total of 29 feedback for
store manager training. , PARTIAL SOURCE: Based on the provided partial sources, we
received a total of 32 feedback for store manager training. , PARTIAL SOURCE: Based on
the provided partial sources, we received a total of 36 feedback for store manager
training. , PARTIAL SOURCE: Based on the provided partial sources, we received a total
of 37 feedback for store manager training. , PARTIAL SOURCE: Based on the provided
partial sources, we received a total of 16 feedback for store manager training. '}
```

Figure 18. *Thinking process without term ”overall” in prompt.*

As can be seen here, AI didn’t anymore try to guess the impossible, but only gave the answer, number of given feedback for each slice of the document that was under processing at a time. Then, counting all the sub-sums together, it yielded the correct answer, 150 feedback. This is exactly what is expected. Due to this kind of challenges and via analyzing this “thought process”, attempts to introduce context awareness into code was made. This concept was decided to be called as “**base question**”. ›› **RESULT2**

4.2.2 Base Question, First Attempt

The idea of the base question was simple, remove extra specifiers from the question, to avoid confusing the AI like explained in previous sub-chapter. For instance, if asking “How many feedback did we receive overall for the team leader training?”, the base question would be “How many feedback did we receive?”. Thinking here was to automate the simplification of the question when processing the parts, no matter in which format user made the question.

The approach tried at this stage was to give rather elaborated explanation and samples to AI, and then ask it to form the base question. Program 7 below records the specific part of the prompt for this purpose:

```
    guidance_for_partial = """ For instance, if question is: How many feedback did we receive
    overall for the team leader training? => Base question: How many feedback did we receive?
Another example for the question: Give the concise summary for worker motivational training.
=> Base question: Give consise summary.
Yet another example of the question: List all the negative feedback for the training => Base
question: List the negative feedback.
In this manner, define the base question on each case separately depending on the actual
question presented. Remove the context entirely, aiming the valid question but without
references to specific event, training, meeting, occasion, or such. However, please notice
that base question can be also more complex. For example question: Assign each feedback to
sentiment category and give me the count of feedback per category. => Base question: give
count of feedback per sentiment category. ALWAYS think through what requestor wants to
understand primarily. For the question: Assign each feedback to sentiment category and give
me the count of feedback per category, YOU CANNOT only say: How many feedback did we
receive?, because this would not answer the question asked: how many feedback per sentiment
category? \n\n
BASE QUESTION: """
```

Program 7. *Phase 2 system prompting for "base question".*

Through several examples (shots) we aimed the AI to "get it" and form a base question out of user's prompt. Elsewhere in the code the guidance shown in Program 7, along with user's prompt, was sent to GPT API to form the base question. Program 8 shows how this was made in a code:

```
# generate the base_question that can be used for parts
user_q = 'Generate the base question, that is context-agnostic FOR THE FOLLOWING:
-- ' + history[-1]["user"] +' -- \n\n' + self.guidance_for_partial + "\n\n"
```

Program 8. *Phase 2 attempt to create relevant base question via ChatGPT API.*

However, this approach was not working well, sometimes not at all. For instance, in a test run when asking a question "*how many feedback did we receive overall?*", the resulted question was "*How many feedback did we receive overall?*" so there was no change at all. This was a clear failure, a prompt slightly deviating from examples given in system prompt did not yield desired outcome.

Not all user input are questions, they can be requests as well. For instance, another user prompt "*assign each feedback to sentiment category and give me the count of feedback per category*", which is not a question at all, the outcome was: "*How many feedback did we receive?*". The thinking process here is totally misguided. Due to these less successful attempts this approach was dropped, to be followed by another try in Phase 3 which was more successful.

4.2.3 Concept of Partial Source and Combination of Partials

As shown earlier, the code used concepts of “partial source”, and “combination of partial sources for final answer”. Despite dropping the first version of the base question, the idea of partials and combination of them continued to live. This section shows how these concepts were made known to AI. These rather elaborative elements of guidance were given via system prompt, here are extracts of the definitions. ›› **RESULT3**

```
Sources you have can be single PARTIAL SOURCE only, which is the fraction of the document
under analysis, or it can be a COMBINATION OF PARTIAL SOURCES FOR FINAL ANSWER, which
contains several PARTIAL SOURCEs under it. ===
```

Program 9. *Extract of system prompt to elaborate existence of partial and full data.*

Program 9 shows the basic definition made to the AI about the two above-mentioned possible types of data. These are single part of the source data, or a list (combination) of parts, meant for generating the final answer that is output to user via UI.

```
In case of single PARTIAL SOURCE, do not try adapt your answer to the earlier conversation
where there seems to be a contradiction: for instance, assistant may have replied that we
have received 304 feedback in total, but this response in conversation history applies to
entire context, not the single PARTIAL SOURCE. In such cases, do not try to force the counts
of partial into the earlier assistant answers. In case of single PARTIAL SOURCE only, do not
get confused with terms like OVERALL or TOTAL, or other terminology that may refer the entire
document. Just answer the MAIN question (for instance: number of feedback) using the PARTIAL
SOURCE you have. In case of PARTIAL SOURCE and numbered feedback, do not use the numbering
when question concerns counts because in such cases the feedback numbering may get mixed with
counts. DO NOT count spaces, newlines, tabulators or other whitespace characters as elements
or objects of concern but skip them in counting. ===
```

Program 10. *Extract of system prompt, aiming to guide the behavior through samples.*

Program 10 shows another part of the prompt, which admittedly very verbose. What is visible in this part of the prompt is the explicit guidance not to count whitespace characters as elements. At this point difficulties getting correct counts were experiences, and it was not clear what it was.

Specific guidance not to count whitespaces as entries was making a positive difference in test runs, though. Now afterwards it is easier to see that something was wrong in the process when transforming the text into PDF. Retrospectively, we found this out later in final documentation phase, as recorded in the analysis part for Phase 3. This is the probable reason for this guidance making an impact, and counts being difficult to system.

```
If you have several PARTIAL SOURCEs following COMBINATION OF PARTIAL SOURCES FOR FINAL
ANSWER, consider multiple PARTIAL SOURCEs as subsums or partial answers for the same question
across the entire document. In such cases create the total answer thinking step-by-step by
combining all the PARTIAL SOURCEs to your final answer. Each PARTIAL SOURCE must contribute
to the final answer, especially if there are counts. DO NOT count spaces, newlines,
tabulators or other whitespace characters as elements or objects of concern but skip them in
counting. Do not derive the final answer based on one PARTIAL SOURCE if several are present.
For EXAMPLE, suppose your sources state like this: COMBINATION OF PARTIAL SOURCES FOR FINAL
ANSWER: PARTIAL SOURCE: We received a total of 29 feedback responses. , PARTIAL SOURCE: Based
on the provided partial sources, we have received a total of 32 feedback. , PARTIAL SOURCE:
Based on the provided partial sources, we have received a total of 36 feedback. => Here you
should consider the several similar answers for the same question as subsums or partial
answers and create the total answer step-by-step by combining all the PARTIAL ANSWERs to your
final answer. In this case, we have received 97 feedback in total. ===
```

Program 11. *Extract of system prompt, further guidance on behavior.*

Finally, Program 11 shows an attempt to steer the behavior into desired direction. This part of the prompt has for instance guidance for summarizing the counts from separate parts together.

4.2.4 Parallelism in Handling the Parts

Compared to the execution of few lines of Python code, response time with OpenAI API appeared to be long, several seconds per request. Unfortunately, exact measurements are not available, but in certain cases even timeout defined by web software was exceeded due to several rounds of iterative calls aiming to reduce the document to proper size for final answer generation. Playing with parameters like window size made some difference, but significant improvement was achieved when calls were made parallel. The idea is to send multiple requests to API almost simultaneously and then wait then to be finished, also almost simultaneously. This worked even with the single core virtual machine. ›› **RESULT4**

To handle the parts of documents (instead of mere pages like in Phase1) method was renamed and adapted, like seen in Program 12 below:

```
async def process_parts(self, history: list[dict[str, str]], overrides: dict[str, Any],
part: str):
    """
    H Heimonen, 2023. Asynchronous method to handle the parts (chucks of string) by
    passing them to the Chat Completion API to complete the prompt.
    Arguments:
    history - list of questions and answers from earlier converstation (within token
    limits)
    overrides - options from UI, used here to override the temperature parameter if
    present, otherwise 0
    part - string to be handle, i.e. the data to base the answer on
    Returns: response from Chat Completion API
    """
    system_message = self.system_message_chat_conversation.format(injected_prompt="",
    follow_up_questions_prompt="")

    messages = self.get_messages_from_history(
        system_message,
        self.chatgpt_model,
        history,
        history[-1]["user"] + "\n\nPARTIAL SOURCES:\n" + part,
        max_tokens=self.chatgpt_token_limit - MAX_RESULT_LENGHT)

    chat_completion = await openai.ChatCompletion.acreate(
        deployment_id=self.chatgpt_deployment,
        model=self.chatgpt_model,
        messages=messages,
        temperature=overrides.get("temperature") or 0.0,
        max_tokens=MAX_RESULT_LENGHT,
        n=1)

    response = chat_completion.choices[0].message.content.strip()
    response = "\nPARTIAL SOURCE:\n" + response + "\n\n"
    response = re.sub('\s+', ' ', response)

    return response
```

Program 12. *Phase 2 method for processing the parts of the text.*

The logic is mostly the same as before in Phase 1, but the essential change here is that looping is not happening within the method itself. Instead, parallel calls are devised elsewhere in the code. Program 13 shows how:

```
while len(full_text) > MAX_NUM_OF_CHARS:
    parts = textwrap.wrap(full_text, PART_SIZE)
    results = []

    # Here we utilize asynchronous coroutines to speed the operations. asyncio.
    gather runs multiple coroutines concurrently by utilizing event loop.
    process_part_with_parameters = partial(self.process_parts, history, overrides)
    tasks = [process_part_with_parameters(part) for part in parts]
    results = await asyncio.gather(*tasks)

    full_text = ",".join(results)
```

Program 13. *Processing of parts via parallel calls.*

In case the first run does not condense the text enough for generating the final answer, i.e., the limit we defined would still exceed, another iteration inside while-loop is taken. Withing the loop, full text is first sliced into parts and stored into list “parts”. Then, multiple coroutines get triggered, adding necessary parameters in, and results are stored into list. Finally, the contents of the list get joined as a continuous text.

4.2.5 Other Aspects for Phase 2

Several aspects from Phase 1 were retained:

- The use of synthetic data was kept the same. In the very beginning the explicit decision was made to use artificial data due to security reasons and focusing on essentials, in terms of POC software.
- Use of Azure search service, combined with reading the whole document based on search result.
- Logging continued to be used.
- Use of modified preparation scripts, including uploading the document to blob without splitting it, because Azure Cognitive Search and indexing was still in use.

4.2.6 Analysis for Phase 2

There were few findings from Phase 2 worth highlighting:

- When injecting the data into prompt, we are effectively reducing, even preventing the possibilities for hallucination. However, when handling data in parts, missing context can mislead the AI and yield incorrect answers that don’t appear to make any sense on surface. ›› **RESULT2**
- Even single word in a prompt can make all the difference. As an example, included “overall” is enough to derail the train in the scenario when handling the parts, if not taken care of. Prompts do matter! ›› **RESULT2**
- Seemingly similar data fragments can yield totally different type of answers (“don’t know” vs “36 feedback”). It was not clear why at this point. There was some indication that something could have been wrong in the process of converting the text into PDF (See note about whitespaces already in Phase 1. Also see topic in Phase 3 analysis.). At some point in the last phase, Phase 3, the process

was changed, and instead of converting MS Word documents to PDFs, text files were used as a source to PDF. Thinking retrospectively, even mere text files could have worked. (Based on the comment from the company this thesis is for text-files worked even more reliably than PDF with the reference design.)

- To build a bridge between parts and the whole, concept of the base question was coined and tried, but rather unsuccessfully at this stage. The idea, however, felt relevant so it was tried again in Phase 3.
- In scenarios where time-consuming IO-operations are triggered and then waited for, parallel coroutines make a lot of sense if true parallelism is possible and things don't have to go in sequence. ›› **RESULT4**

4.3 Phase 3

Last, third phase, got rid of usage of Azure Cognitive Search and related indexing that are fundamental elements in the reference software. They were not seen essential to the success of intended use case. This simplified the implementation greatly.

In addition, new approach to the base question was implemented and it worked better than in previous attempt. Finally, structured system prompt was created, relying on elements injected into prompt. Essential parts of code, including almost all the code changes made for the POC (except some configurations for compiling etc.) can be seen in Appendix D.

4.3.1 Abandoning Azure Cognitive Search and Indexing

Reference architecture was selected due to advantages it gave from the start as explained before. However, implementation in Phase 2 already worked around many of the features and bypassed them. When continuing to try with even longer document (see Appendix E) it was found out that there was a token limit associated into indexing as well. As explained already earlier, in Phase 1 the document was indexed as a single entity without splitting it. Now with even larger document used during the development the token limit associated with indexing was hit and execution stopped with the error message.

As there really was not much use for the search and indexing in the POC software, decision was made to get rid of it altogether. Use case is different when analyzing customer data: it can be expected that operator knows what he/she wants to analyze and does not have to rely on search to find it.

This decision meant that actual code changes were needed only in couple of places, and almost all of these were for class “ChatReadRetrieveReadApproach”. The implementation for document indexing as well as custom document upload were dropped, for instance. The only code change remaining outside of ChatReadRetrieveReadApproach class was the injection of blob client and Azure storage container references as explained earlier. This remained similar as before.

In “ChatReadRetrieveReadApproach” class dropping the Azure Cognitive Search and indexing meant a lot of simplification. The selected approach was that the document to analyze is stored as “input.pdf” into blog storage. Then without any searching this is read and processed. Such a static approach is of course not proper approach in fully productized version but that was not the intention, either.

At this stage Microsoft had changed the implementation to support either streaming or non-streaming version. Method called “run_until_final_call” was acting as a “main” function in both cases, only working a bit differently. As before, some elements of Microsoft code were re-used as-is and then method adapted for POC. Program 14 below shows the beginning of the method “run_until_final_call”:

```
async def run_until_final_call(
    self,
    history: list[dict[str, str]],
    overrides: dict[str, Any],
    auth_claims: dict[str, Any],
    should_stream: bool = False,
) -> tuple:

    original_user_query = history[-1]["content"]
    results = []     # used to store thought process
    filename = "input.pdf"
    pages = []

    async for page in self.get_pdf_doc_from_blob(filename, self.storage_account, self.
    storage_container):
        pages.append(page.extract_text())
    full_text = ",".join(pages)
```

Program 14. *Revised method (by Microsoft), modified for purpose (by author). This part of code orchestrates the overall process.*

Here can be seen how hard-coded value of filename is set, and then utilized in reading the content from the blob storage. Pages are joined and assigned to “full_text” which now has all the text in document.

4.3.2 Base Question, Second Attempt

Earlier attempt to get the base question working has been explained inside Phase 2, and theory and thinking behind that will not be repeated. Now new approach was tried, however. Program 15 presents the dedicated method to create the base question:

```
async def create_base_question(self, question: str):
    """
    H Heimonen, 2023. Support method that is used to extract "base question" from
    potentially more elaborated question.
    Arguments:
    question - actual question asked by user
    """
    base_question_request = "Generate the base question or request from the prompt. Do
    this by simplifying the question or request, by removing the entire object from the
    sentence if it is present, or only the object modifier if the question would not
    make sense when removing whole object. Also, remove terms like 'whole', 'overall'
    and so on that refer to full context. For instance: Question: Provide concise
    summary for 7 months store manager training feedback. Base question: Provide concise
    summary for feedback. Another example: Question: How many participants there is
    overall? Base question: How many participants there is? \nQuestion: "  + question +
    "\n\nBase question: "

    messages = self.get_messages_from_history(
        "",
        self.chatgpt_model,
        [],
        base_question_request,
        max_tokens=self.chatgpt_token_limit - len(base_question_request))

    chat_completion = await openai.ChatCompletion.acreate(
        deployment_id=self.chatgpt_deployment,
        model=self.chatgpt_model,
        messages=messages,
        temperature=0.0,
        max_tokens=MAX_RESULT_LENGHT,
        n=1)

    base_question = chat_completion.choices[0].message.content.strip()
    return base_question
```

Program 15. *Revised Phase 3 method for creating "base question".*

The method includes the definition for the actual request "system prompt", but of course that could be defined outside the method as well. The approach used this time, as defined by "base_question_request", is to use linguistic approach. We instruct that base question can be generated **by removing either the entire object from sentence** if it is present, **or only the object modifier** if the question would become irrelevant by removing the entire object. To define this approach, a bit support by ChatGPT was used in the thinking process, like hinted earlier. In addition, terms already known to be confusing when handling parts of document, i.e., terms like “whole” or “total” were explicitly instructed to be removed. To increase chances for success couple of examples are given. Following Figure 19 defines the desired behavior through example:

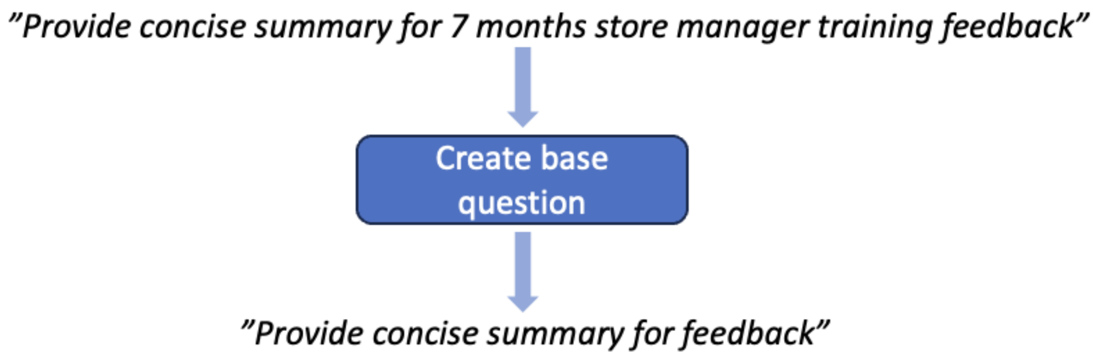


Figure 19. *Conceptual example of forming the base question from the prompt by removing the object modifier.*

Figure 19 shows the concept of base question through example. This is the same behavior that was instructed in system prompt that is part of “*create_base_question*” method. Method gets called before proceeding into handling the parts, to define proper base question.

Thinking back, more careful consideration for the cases which are not actual questions at all could have been required. In addition, term “feedback” can be seen belonging to object modifier, but as examples are given it looks that the approach works. To make it bulletproof more testing and iteration would be required! But despite of these limitations, now the cases that did not work earlier, worked much better as shown by Figure 20:

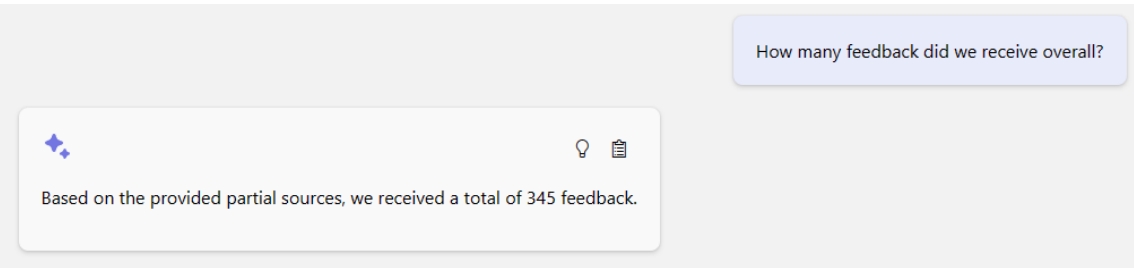


Figure 20. *Example run including the previously problematic ”overall” term.*

Here we get quite good answer against the data used (Appendix E), even though not fully correct (exact number is 365). Anomalies and issues related to counts remained somewhat loose end almost until the completion of the writing process of this document. During the documentation phase it was learned that something had gone wrong when converting the text documents to PDFs, few items were together instead of on their own rows. Please see the last comment in the sub-chapter collecting the analysis for Phase 3.

What is more important here than exact count, however, is to see what was the base question that was formed, and consequently used when handling the parts. Figure 21 has a screenshot from the log stream where the formed base question was printed:

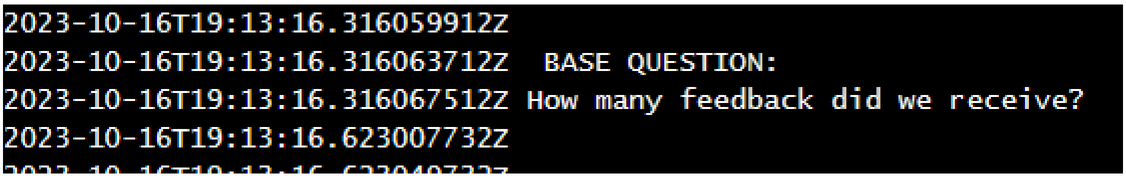


Figure 21. *Log stream revealing the created base question.*

Base question was correctly defined. However, it can be argued that getting this question for counts correctly is not a breakthrough, as guidance to remove terms “overall” was quite explicit in given prompt. On the other hand, similar explicit guidance did not work before as was explained in Phase 2, but now it did. Figure 22 holds another example:

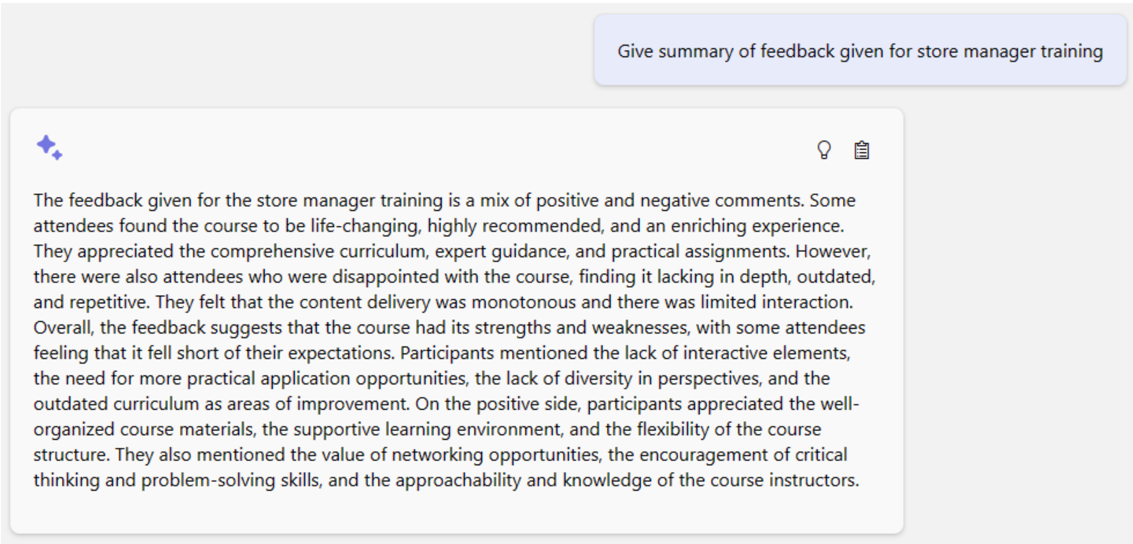


Figure 22. *Sample run with another prompt, and without term ”overall”.*

Figure 22 shows a request, followed by the summary. Again, from the log stream displayed by Figure 23 we can see that proper base question was created:

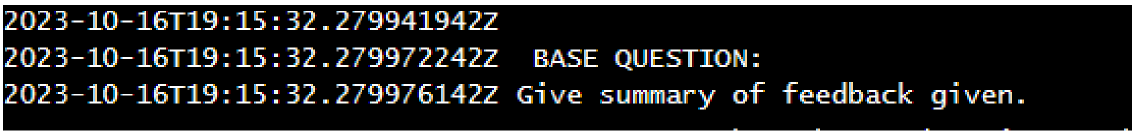


Figure 23. *Log stream output for created base question, in context of another sample run in Figure 22.*

This is what we want, and for this there was no explicit guidance given in prompt, meaning the logic worked ok. This kind of base questions are less prone to confuse AI in a way that was explained in Phase 2. ›› **RESULT5**

Let's summarize the Design Theory for base question through Figure 24:

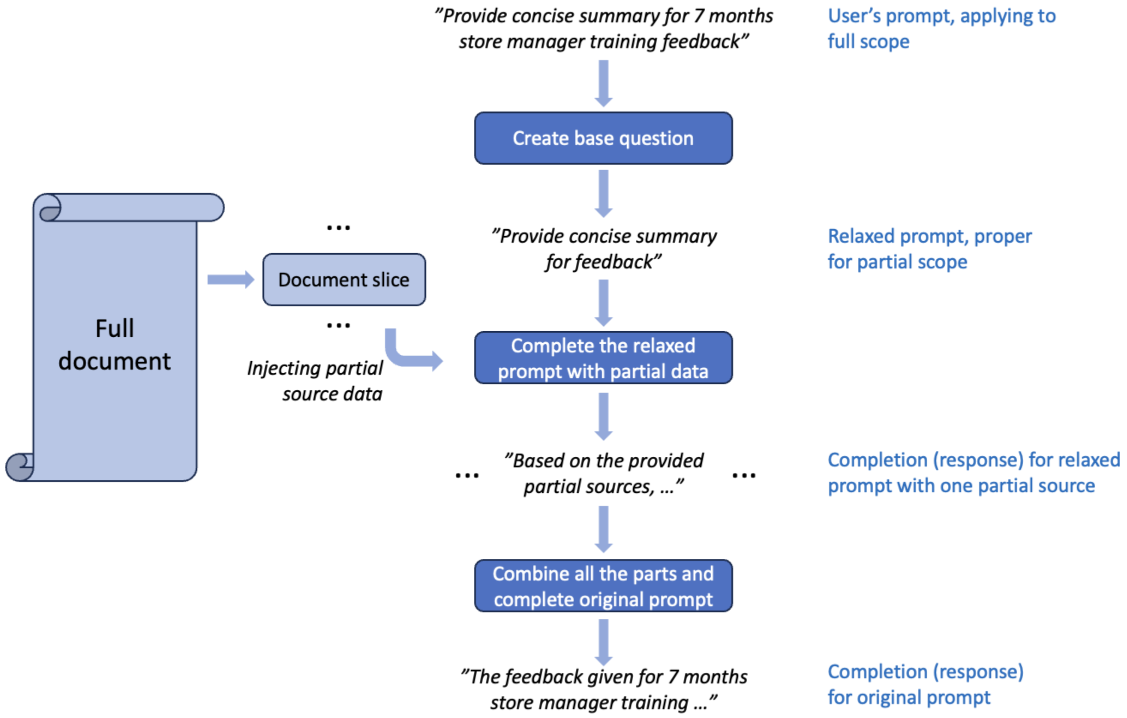


Figure 24. *Theory of base question.*

Figure 24 depicts the full theory and how the concept of base question works across the entire workflow. User's prompt, relevant to full scope, gets processed by relaxing it as described before. This base question becomes a new intermediate prompt that is used with document slices. AI is less likely to get confused. After all the parts have been handled, they are combined, and initial prompt gets completed. ›› **RESULT5**

Outside of the base question itself couple of things are noteworthy in the Figure 22 showing the conversation with the AI: in the middle of the response we can see the term "Overall" which seems to be what ChatGPT often uses as a first word in closing paragraph. However, here the story continues after that, and knowing the background it seems that we can still somehow perceive the invisible borderline between separate parts that were used to generate shows summary of summaries. This could still be improved. ›› **RESULT3**

Another thing is that while the summary is correct, comparing to the contents of Appendix E, the beginning of the document is heavily emphasized. Sometimes this might be ok, but more effort could be put how to balance the summarizing, e.g., via system prompt. ›› **RESULT3**

4.3.3 Structured System Prompt

In Phase 2 the system prompt was quite long set of directives that can get confusing. In this last phase the modular approach to system prompt was used, following the same idea as was used for instance for follow-up questions or injected prompt in the baseline reference software. ›› **RESULT6**

In this approach certain categories relevant to the goal of POC software were introduced, but in no means are they all-embracing. Content of the injected parts is more or less the same as earlier (in Phase 2), but more categories and parts could be easily added. Program 16 depicts the categories used now:

```
configured_system_message = """
{basic_behavior}
{in_context_learning}
{data_types}
{context_awareness_partial}
{partial_source_and_counts}
{context_awareness_full}
{summarizing}
{sentiment_analysis}
{sentiment_and_counts}
```

Program 16. *Defined categories for structured system prompt.*

Let’s shortly open the categories listed in Program 16:

- Basic behavior (Program 17):

```
# Basic behavior

basic_behavior_assistant = """Assistant helps with various questions related to source
documents. Be brief and concise in your answers. Do not repeat the question in your
answers.
"""

basic_behavior_elaborate = """Assistant helps with various questions related to source
documents. Be elaborate but accurate in your answers.
"""
```

Program 17. *Alternative prompt options for basic behavior.*

This is meant to define the basic approach for the assistant AI. Couple of examples were given in code: first option in Program 17 is what was used earlier, including “brief and concise" answers. Second option requests elaborate answers. Of course, any other things could be added, e.g. hostile or overly polite attitude if those for some reason that were required.

- In-context learning (Program 18):

```
# In-context-learning

in_context_learning_injected_data = """Answer ONLY with the facts listed in the list of
sources. Do not generate answers that do not use the sources below. If asking a
clarifying question to the user would help, ask the question. If there is not enough
information in sources, say you do not know.
"""
```

Program 18. *Guidance to use only given source data.*

Program 18 shows how we guide to answer based on provided, injected data only. No alternatives are given here as this is vital for RAG-pattern anyway.

- Data types (Program 19):

```
# Data sources

data_types_partial_n_full = """Sources you have can be single PARTIAL SOURCE only, which
is the fraction of the document under analysis, or it can be a COMBINATION OF PARTIAL
SOURCES FOR FINAL ANSWER, which contains several PARTIAL SOURCEs under it.
"""
```

Program 19. *Guidance for two types of data, partial and full.*

Concepts of partial and full data source types seen in Program 19 have been already explained in detail earlier, please refer to that explanation. No new logic was introduced here.

- Context-awareness for partials (Program 20):

```
# Context awareness for partials (fragments of a document)

context_awareness_partial_basic = """In case of single PARTIAL SOURCE, do not try adapt
your answer to the earlier conversation if there seems to be a contradiction: for
instance, conversation history may indicate that we had got 500 items in total, but this
response applies to entire context, not the single PARTIAL SOURCE. In such cases, do not
try to force the counts of partial into the earlier assistant answers.
"""
```

Program 20. *Guidance for context awareness, in case of partials.*

Single basic option is available in code, this retains the logic already explained earlier. Guidance shown in Program 20 aims to ensure that partial documents are handled without trying to force the answers to match the conversation history. For instance, if conversation history states that there are 500 feedback, partial would not have as many.

- Partial source and counts (Program 21):

```
# Handling partial sources in case of counts (fragments of a document)

partial_source_and_counts_basic = """If PARTIAL SOURCE includes numbered items or lines
and question concerns counts, DO NOT include these order numbers into answer, unless
spesifically requested, as these order numbers may get accidentally counted into actual
count subsums. DO NOT count spaces, newlines, tabulators or other whitespace characters
as elements or objects of concern but skip them in counting. When answering based on
PARTIAL SOURCE try not to repeat the term partial source in the answer.
"""
```

Program 21. *Specific guidance on counts, in case of partials.*

Only single basic option has been provided with "_basic" extension, shows in Program 21. This enhances the previous segment, dedicated for counts as those appeared to be challenging (see analysis chapter for Phase 3, though). This is again re-using the logic opened up earlier, no changes.

- Context awareness for combination of partials (Program 22):

```
# Context awareness for summary (multiple partials)

context_awareness_full_basic = """If you have several PARTIAL SOURCEs following the
title COMBINATION OF PARTIAL SOURCES FOR FINAL ANSWER, consider multiple PARTIAL SOURCEs
as subsums or partial answers for the same question across the entire document. In such
cases create the final answer thinking step-by-step by combining ALL the PARTIAL SOURCEs
to your final answer. Each PARTIAL SOURCE MUST CONTRIBUTE to the final answer,
ESPECIALLY if there are counts: if question is about counts, ensure step-by-step that
ALL the PARTIAL SOURCEs are summed together, but only show the final sum, do not show
the calculation process! Do not derive the final answer based on one PARTIAL SOURCE if
several are present, all PARTIAL SOURCEs must contribute the final answer. However, aim
for balanced and fluent final answer that does not make it obvious that it has been
combined from several partial sources. If question is about counts, include ALL the
PARTIAL SOURCEs step-by-step. Be concise and relatively short when forming the final
answers, try not to repeat same things in combined final answer.
"""
```

Program 22. *Guidance for combination of partials.*

Guidance shown in Program 22 has been opened earlier, it defines what to do in case of having a combination of partials. Here also only single option has been provided.

- Summarizing (Program 23):

```
# Behavior for summarizing

summarizing_concise = """If asked to summarize without extra guidance, give a concise
summary of the available content."""

summarizing_elaborate = """If asked to summarize without extra guidance, give elaborate,
yet accurate summary of the available content."""
```

Program 23. *Guidance for summarizing, two alternative approaches.*

Program 23 shows the definition for the behavior when summarizing things. In here two alternative options were given, either concise or elaborate.

- Sentiment analysis (Program 24):

```
# Behavior for analyzing sentiments

sentiment_analysis_basic = """If asked anything about the sentiments or sentiment
categories, always think through step-by-step and ensure you assign single sentiment for
each individual comment or feedback. First check step-by-step how many elements, e.g.
feedback, you have in a source, and then handle each individual element, for instance
feedback, as a whole even if it consisted of several sentences. It is NOT allowed to
split single element into several sentiment categories, unless specifically asked for.
For instance, if the source includes 20 separate elements, for instance feedback, your
should end up with 20 sentiments assigned. However, do not assing sentiment to sentences
or paragraphs that are clearly not feedback but some common explanation or structure for
the document. DO NOT count spaces, newlines, tabulators or other whitespace characters
as elements or objects of concern but skip them in counting. Also keep in mind what is
said before about handling PARTIAL SOURCE and COMBINATION OF PARTIAL SOURCES FOR FINAL
ANSWER.
"""
```

Program 24. *Guidance for analyzing sentiments.*

As seen in Program 24 this is a bit more elaborate guidance, despite being called “basic”. This suffix is mainly there in case we would like to introduce some other option. One thing here was the guidance not to start splitting longer feedback to multiple sentiments, unless specifically asked for.

- Sentiments and counts (Program 25):

```
# Guidance for sentiment analysis and combined counts

sentiment_and_counts_basic = """If PARTIAL SOURCE includes numbered items or lines and
question concerns the count of sentiments, sentiment categories, or like, DO NOT include
these order numbers into answer, unless spesifically requested for, because these order
numbers get easily counted into actual final count which is not the intention. BUT
STILL, EACH numbered feedback MUST BE assigned a sentiment step-by-step and counted in.
If question concerns counts of sentiments, DO NOT list the items or any intermediate
results or thought process for categorization, but give the COUNTS ONLY! Think
step-by-step and give the answer for counts, without showing the thought process or
individual elements of feedback.
"""
```

Program 25. *Specific guidance regarding the counts and sentiments.*

Program 25 displays a little bit of special case already. This was added when it was noticed that when AI started feeding back or repeating the possible order numbers of feedback items in its response, they started confusing the counts calculations badly.

Finally, usage of structured prompt is mostly basic python, selected options are injected and this defines the actual system prompt. Example shown in Program 26:

```
def configure_system_message(self):
    """
    H Heimonen, 2023. Support method for configuring the system message, in other words
    the AI behavior.
    """
    message = self.configured_system_message.format(
        basic_behavior=self.basic_behavior_assistant,
        in_context_learning=self.in_context_learning_injected_data,
        data_types=self.data_types_partial_n_full,
        context_awareness_partial=self.context_awareness_partial_basic,
        partial_source_and_counts=self.partial_source_and_counts_basic,
        context_awareness_full=self.context_awareness_full_basic,
        summarizing=self.summarizing_concise,
        sentiment_analysis=self.sentiment_analysis_basic,
        sentiment_and_counts=self.sentiment_and_counts_basic
    )
    return message
```

Program 26. *Configuring structured system prompt in code.*

Here the basic behavior category the normal assistant has been selected to get concise answer. We could have selected the elaborate version in here. Same approach applies to all categories, many more can be defined if seen fit.

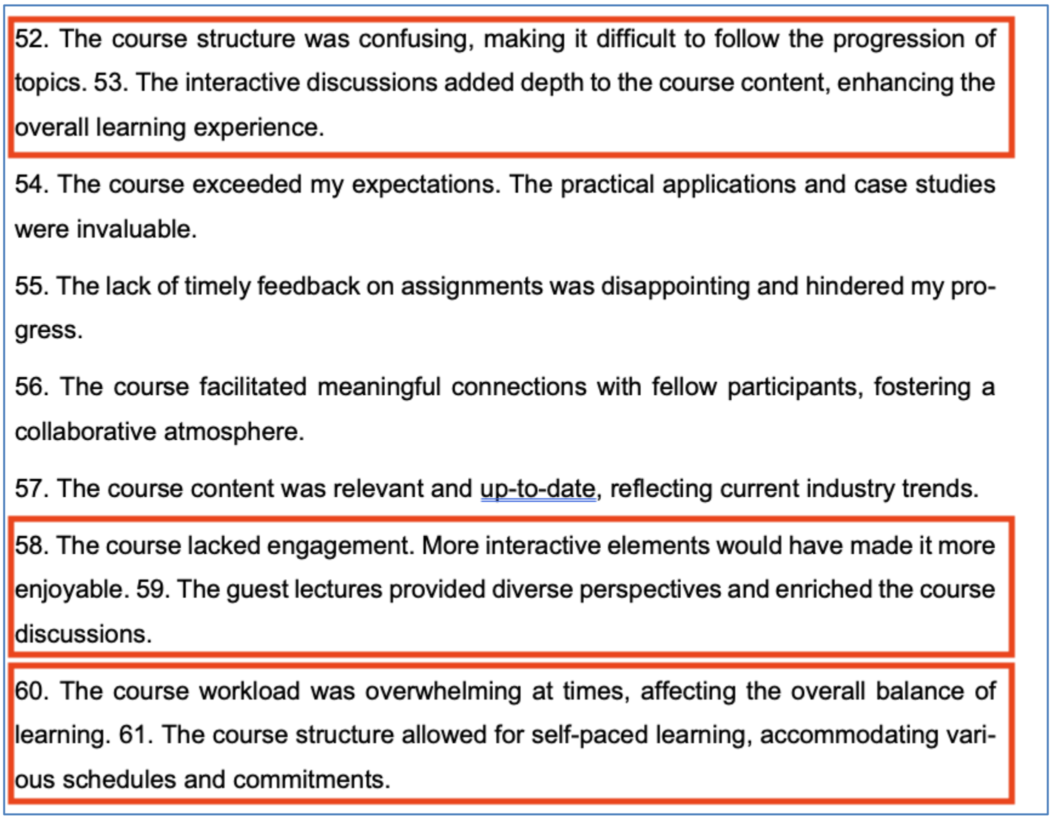
Overall, the prompt is currently very long, partially because of detailed guidance as trying to work around with the difficulties experienced with PDF documents and counts. As a future improvement topic it could be studied how to simplify it.

4.3.4 Analysis for Phase 3

Certain essential topics from Phase 3 are elaborated here.

- Abandoning Azure Cognitive Search and indexing greatly simplified the software. This could have been done earlier but was only implemented in the last phase. The current implementation uses hard-coded approach for picking particular file from blob-storage, but like mentioned earlier this document this is not production ready solution. In productized version getting the list of available documents in blob storage and selecting a filename from the dropdown menu would be one possible approach.
- As explained earlier, references to full context, when full context is not available, can cause AI to get confused and create incorrect answers. Introducing the (possibly novel) idea of “base question” worked better with this second attempt by using the different approach. As context window sizes of new GPT versions keep rapidly growing, this idea has less use in normal scenarios, admittedly. Also, more studying and thorough testing would be required to gain more confidence on the process in various scenarios. Basic idea seems to work, but with the current implementation it may be easy to find a question posed by user that confuses not the AI, but the base question generation itself. This was added in the late phase of implementation and testing is not extensive. However, on some level the idea of base question could be considered even mid-range Design Theory, including Models and Constructs. Regarding the concept of “mid-range” in the context of design theories, especially in case of ISDT (Information Systems Design Theory), Gregor (2006) states that mid-rage theories are “moderately abstract and limited in scope.”. (Gregor, 2006, according to Vaishnavi and Kuechler, 2015, p. 69).
- Structure prompt didn’t bring new design knowledge as such but was created to bring some order to already extremely long system prompt. It also gives simple mechanism to configure different versions of system prompt in organized way. At the moment the configuration is static, but nothing prevents in future implementing the configuration part via environment variables for instance which makes it dynamic without needed code changes.
- Even in the end of the last phase some open questions remained regarding the difficulty in dealing with counts. The idea was that this had something to do with text to PDF conversion, but that could not be confirmed. Still, request not to count

whitespace-characters seemed to impact the results so it was clear something was off with the data. Only at the point when finalizing this document more clues were found: when copy-pasting the text content from the last version of generated PDF-document, “input.pdf” to Appendix E, it was found out that indeed something was wrong. In few cases, couple of numbered comments were not separated but instead were somehow concatenated together as seen in Figure 25:



52. The course structure was confusing, making it difficult to follow the progression of topics. 53. The interactive discussions added depth to the course content, enhancing the overall learning experience.

54. The course exceeded my expectations. The practical applications and case studies were invaluable.

55. The lack of timely feedback on assignments was disappointing and hindered my progress.

56. The course facilitated meaningful connections with fellow participants, fostering a collaborative atmosphere.

57. The course content was relevant and up-to-date, reflecting current industry trends.

58. The course lacked engagement. More interactive elements would have made it more enjoyable. 59. The guest lectures provided diverse perspectives and enriched the course discussions.

60. The course workload was overwhelming at times, affecting the overall balance of learning. 61. The course structure allowed for self-paced learning, accommodating various schedules and commitments.

Figure 25. *Evidence of things being wrong in text > PDF conversion. It looks like that there are issues with whitespace characters.*

In other parts of the document the formatting was even worse. Now when copy-pasting the same text from the plain text file (file that was used as origin for conversion), no such thing happened for the feedback. This seems to confirm that something had gone wrong when converting the txt-format to PDF-format using Notepad++. These anomalies were not even in the places of page breaks in the source “input.pdf”, but in apparently random locations. This almost certainly explains the issues with counts, two separate feedback were counted as one in those cases. There were also other issues, like single feedback breaking apart to several lines, etc. Using a mere text file would therefore have been the safest way forward.

4.4 Some Alternatives and Recent Development

All the previous sub-chapters describe what was done as a part of the POC software. Like indicated few times, however, the development of the field has been fast. While implementing the POC and writing this document, things have changed already multiple times.

This sub-chapter opens some of the alternatives that have become available. The point of view emphasizes the goal of the thesis, i.e., handling long documents to which user can make questions. The direct technical application of the POC software, which is splitting the documents to parts, starts to get greatly obsolete due to latest announcements.

4.4.1 Microsoft Bing a.k.a. Copilot

Microsoft's AI-powered Bing search along with new Edge browser were announced during early 2023 ("Reinventing search", Mehdi, 2023a). Included AI add-on in Edge browser was recently re-named as "Copilot", and Copilot is getting integrated into other Microsoft 365 Office products as well ("Announcing Microsoft Copilot", Mehdi, 2023b; "Our vision to bring Microsoft Copilot to everyone", 2023). Copilot includes the capability to summarize documents. It seems challenging to find exact information on related token limits, but one Microsoft source discussing on Copilot for MS Word it is mentioned that "Copilot is currently limited to a maximum of around 18,000 to 20,000 words it can process for a single query or prompt for features like generating document summaries and chatting with Copilot.", at the time of accessing this website. ("Create a summary of your document with Copilot," 2023).

When trying Bing in Edge browser during September with the same PDF document that was used during that time for the POC software, results were not good yet. See Figure 26 for the attempted sentiment analysis:

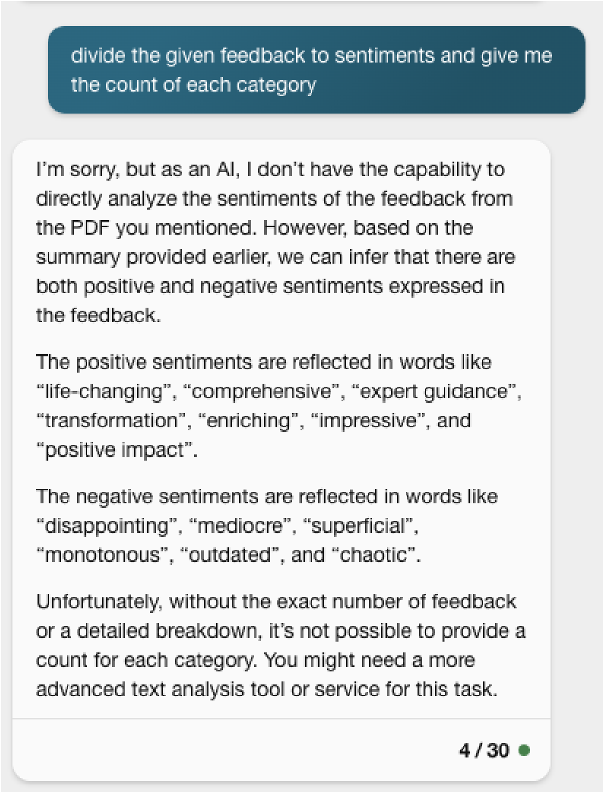


Figure 26. *Attempting to get MS Bing to analyze sentiments from provided PDF.*

Even though Bing was able to detect the document, it "refused" to categorize the sentiments out of the box. It is possible that with more careful guidance, or prompting, better results would have achieved. But if that kind of guidance were required every time for similar use case that is not very handy.

The previous comment that the exact count of feedback is not known, calls for checking what then is the count, in Bing's opinion, shown in Figure 27 below:

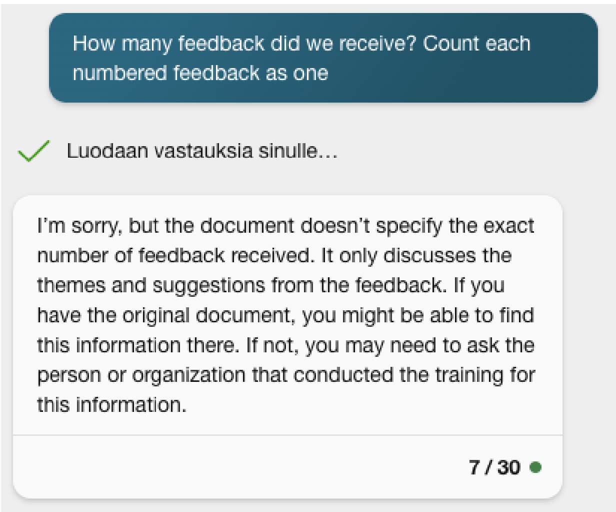


Figure 27. *MS Bing refusing to count the number of feedback in PDF.*

Despite being a bit more specific this time, i.e., what counts as one feedback, Bing didn't count the number of feedback but provided some generic advice how to find out the correct result.

This most certainly is not a full representation of Bing capabilities, however no more effort was used to this investigation.

4.4.2 ChatGPT

OpenAI's web-based chat service has received the capability to accept uploaded documents and analyze or summarize them in various ways. OpenAI's article mentions the current availability for ChatGPT Plus (paid version) and ChatGPT Enterprise users, and feature going to be available soon for API users also. ("File uploads with GPTs and Advanced Data Analysis", n.d.). According to mentioned article the feature supports "All common file extensions for text files, spreadsheets, presentations, and documents.". What is notable is the comment about the token limit per file: "All text text and document files uploaded to a GPT or to a ChatGPT conversation are capped at 2M tokens per files. This limitation does not apply to spreadsheets." ("File uploads with GPTs and Advanced Data Analysis", n.d.). This is quite a leap compared to the limit dealt with during this study, which was 4096 tokens.

At their DevDay 6th Nov 2023 OpenAI announced more features. Along with new GPT-4 Turbo and updated GPT-3.5 Turbo, even more functionality was revealed. Most notable announcement in terms of the scope of this thesis was new "Assistants API" which was mentioned to have "persistent and infinitely long threads" (OpenAI, 2023; "Assistants API," n.d.). Assistants API can be used to build AI assistant that is included into one's own application. How exactly the "infinite" size of Assistants API thread translates to use case of summarizing documents, not sure. However, page for the current Beta version overview mentions that API takes care that requests will always fit the context window. It seems to be using internal optimizations to do that: "Threads don't have a size limit. You can add as many Messages as you want to a Thread. The Assistant will ensure that requests to the model fit within the maximum context window, using relevant optimization techniques such as truncation which we have tested extensively with ChatGPT." ("Assistants API," n.d.). ("Assistants API," n.d.). On surface it sounds that something similar what has been done with the POC software in terms of compressing and summarizing content is happening within the Assistants API implementation.

4.4.3 Amazon Q

29th November 2023 – which is the same day when writing this - Amazon announced its ”generative AI assistant” to public (“AWS Announces Amazon Q”, 2023). It is very early days for Amazon Q but notably one of the features available is “summarizing documents” (“AWS Announces Amazon Q”, 2023).

Press release mentions the capability to connect Amazon Q with customer’s own data and systems, also via 40 built-in connectors for various data sources like Amazon S3, Google Drive, Microsoft 365 and so on. Once information has been synthesized customer can let Amazon Q to generate assistant web application to which users can access using their existing authentication. (“AWS Announces Amazon”, 2023).

5. SUMMARY OF RESULTS

This chapter compresses the earlier effort from two previous chapters and summarizes the results. Earlier mentioned details will not be repeated any more but only shortly referred to. For deeper understanding of detailed results and to see how they were derived, reader is strongly encouraged to familiarize with chapters about design science research methodology and system description. This approach has been selected as describing results in detail make more sense to reader when context is immediately available. Whenever a paragraph has contributed to one of the results listed in this chapter, the corresponding result identification number has been listed alongside the paragraph, in this format: ›› **RESULT1**. This, for instance, would mean that the immediate text context does contribute to RESULT1.

5.1 Collection of Results from System Iterations and DSR Narrative

Results have been divided in two main domains, which reflect the focus areas of the thesis. First focus area is Proof-of-Concept software itself and the results that have come from iterations, towards the research questions. Second area is the DSR methodology, and results coming from analyzing this process via narrative strategy. Not pure by-the-book DSR was used like already mentioned, rather DSR-IS, but still approach that is close enough to justify the analysis and investigation for the process itself.

Within these two main domains, findings have been classified into various types or categories, according to what has been described in earlier chapter about DSR knowledge. Some of the classification could be argued with, but best effort for the categorization has been made. First, we list the results related to Proof-of-Concept software and iterations in Table 4:

Table 4. *Summary of results towards Research Questions.*

DSR Knowledge Category	Result ID, Result Summary
Instantiations	**RESULT1**: Regarding the research questions, LLMs can be used to create grounded summaries and extract actionable insights and sentiments from textual data when injecting it as an input (RAG-pattern) and guiding the behavior of AI via system prompt. Results received from AI resonate well with to the source data used as shown earlier. Proof-of-Concept software uses approach where data is injected into prompt and responses are generated only based on that, preventing hallucinations that generative AI is also known for. Selected reference design benefits from all conversational capabilities of ChatGPT while allowing discussions about own data e.g. in Finnish, in English and so on.
Design Principle	**RESULT2**: When analyzing partial sources (fragments of document) missing context can cause unexpected issues, for instance “cannot tell” - type of answers. In such scenarios, even single word in a prompt does matter. For instance, use of term “overall” can confuse AI, when it does not know it all, at once – and is aware of that. More detailed analysis is available earlier in this document.
Constructs, Instantiations	**RESULT3**: Analysis of large text entities that exceed the size of available context window was implemented via recursive processing of parts and generating summary of summaries. To implement this, two concept were introduced and used in system prompt: 1. Partial source (fragment of the document) 2. Combination of parts (for generating the final summary) As noted in the chapter “Base question, second attempt” the border of the parts can sometimes still be visible and could be improved. Also, it seems that the beginning of used data is getting most weight.
Methods	**RESULT4**: When sequential approach of handling the data is not needed, use of parallelism can give advantages. Waiting times of GTP API are relatively long in computing scene, and POC software utilized parallel coroutines. Previously mentioned concepts (partial source, base question, and combination of parts) made that possible. **RESULT6**: Injection of various behavior, e.g. guidance for sentiment analysis, was implemented to simplify the building process of lengthy system prompt. This structured system prompt gets configured at runtime, but still relies on static selection in code now, not external environment variables. This could be future enhancement.
(Mid-range) Design Theory, including Methods and Constructs	**RESULT5**: To prevent above-mentioned confusion caused by missing full context (when handling the fragment of the document), new mid-range Design Theory called “base question” was created. Base question approach “relaxes” the user prompt so that correct result can be generated, instead of rather impudent and by-the-letter answer “cannot tell based on partial data”. Base question relies on ChatGPT and related specific prompting to generate the relaxed base question from user’s question.

Results derived from the implementation of the POC software and presented in Table 4 answer to research questions defined in the beginning of the study. Regarding the nature of findings, proposed “Base Question” approach can qualify as mid-range Design Theory on some level, even though further study is still needed. Whether the maturity of the proposal is on needed level needs to be discussed. Base question theory could be useful in scenarios where size of the context window forces to split the data into parts. Other results belong to lower design knowledge hierarchies.

Next, results derived via narrative strategy that was applied to the DSR process itself are listed. Table 5 below collects the results from the analysis of the DSR:

Table 5. *Summary of results for design science research process findings.*

DSR Knowledge Category	Result Summary
N/A	**DSR_RESULT1**: In software development some aspects of the operating environment (e.g., shared libraries, API implementations and reference design) are subject to external changes that the owner of a particular DSR effort cannot control. These changes can create disruptions to development process. **DSR_RESULT3**: It is suggested that if a particular iteration of DSR cycle is obviously not yielding any new design science knowledge some formalities would be skipped.
Design Principle	**DSR_RESULT2**: When both solution and problem domain maturities are low, which is still the case with generative AI, concept of “existing artifact” or “state-of-the-art” can be difficult to define in a project that extends for instance across several months. New things are introduced all the time, and today’s news is no more news tomorrow. **DSR_RESULT4**: Traditional DSR cycle could benefit from adding elements of agile into process and making them visible. Some version of Agile DSR has been suggested via literature also. **DSR_RESULT5**: Similarly, as with DSR_RESULT2, intermediate design science knowledge can get old during the project if both solution and problem domain maturities are low. This is the case especially if the knowledge is on the area of technical instantiations and implementation, instead of higher-level paradigms like Design Theory. **DSR_RESULT6**: Loose ends may get clarified even at very late state of the process, for instance when documenting the results.

When observing the Design Science Research process itself, few Design Principle findings were made, as listed in Table 5 above. As generative AI is currently rapidly advancing field, it was interesting to see how quickly something “new” became “old”. That of course is not the case so much when the research field is more mature. In addition, it seems that some added aspects of agile thinking could benefit the DSR, by bringing some flexibility into it, when due. Agile is also the way that most of the software engineers nowadays primarily think, based on my personal experience. Couple of findings did not belong to obvious categories, but they are listed, nevertheless.

5.2 Key Findings for Research Questions

Results in this sub-chapter are based on previous information and recorded results, and background is therefore no more repeated. Intention is to give a focused view of results versus the research questions. Initial, high-level research question was the following:

- **How to use Large Language Models (LLM) effectively in complex business analytics?**

This overarching theme is very wide, though, and it was reduced towards three and more tangible sub-questions that are relevant to actual business scenario and use case. Those three questions will be handled next.

- **Q1: How can artificial intelligence be used to create summaries and extract actionable insight from textual data?**

➔ AI, LLM specifically, can create summaries and extract actionable insight from textual data using the pattern where data is injected into system prompt, and AI is guided to use that data only as a basis for answers. This is known as “RAG” pattern. (**RESULT1**)

➔ RAG-pattern is a good way as it enables combining company’s own data with capabilities of pre-trained AI without fine-tuning of AI model. It can handle ever-changing datasets and use multiple languages. (**RESULT1**)

➔ With POC-software, summaries are received as a response to user’s prompt, which are textual, “conversational” messages provided via chat-like interface. Received summaries and insights seem relevant compared to the provided data, therefore this is feasible approach for business to reduce the human workload when analyzing large texts. (**RESULT1**)

➔ If custom approach, meaning specific AI behavior, is needed, that is controlled via hidden part of the prompt, system prompt. (**RESULT1**, **RESULT6**)

➔ To work around the LLM context size limitation, splitting of text into parts and recursive handling can be used. This is supported by two concepts introduced in this study, which are “partial source (fragment of the document)” and “combination of parts (for generating the final summary)”. (**RESULT3**)

➔ When handling partial data, AI can get confused by questions concerning full context. To mitigate this, new mid-range design theory called “base question” was created to support handling of partial data. (**RESULT2**, **RESULT5**)

- **Q2: How can AI be used to detect sentiments from the text?**

➔ Results for Q2 build on the results of Q1. To detect sentiments from text, AI is guided to use injected data as a source with RAG-pattern, which gives capabilities of pre-trained AI to user’s disposal. This gives a good starting point for sentiment detection. (**RESULT1**)

➔ User can make questions regarding the sentiments in data. Interpreting sentiments is subjective, though. Like with Q1, system prompt can be used to guide the behavior. With ChatGPT the sentiment detection was working reasonably well. (**RESULT1**, **RESULT6**)

- **Q3: What kind of technical architecture and implementation would answer the need?**

➔ Results for Q3 are a collection of results presented for Q1 and Q3. Repeating them shortly, RAG-pattern combining the benefits of pre-trained AI and own data is a good way forward. Longer texts can be handled recursively in parts. Handling of partial data is supported by introduced concepts of partial data, combination of parts, and base question. (**RESULT1**, **RESULT3**, **RESULT5**)

➔ Parallelism can be used to improve performance. Use case and POC-implementation allows handling of parts concurrently. (**RESULT4**)

6. DISCUSSION AND CONCLUSIONS

Main goal of the study was to build a Proof-of-Concept software, and through that process, to derive ideas how can generative AI be used with textual data to generate summaries, give insights, and extract sentiments from that data. Process was guided by defined research questions. Results show that it is indeed possible to use own data together with generative AI and get reasonable results out of it. During last few months the possibilities have increased.

When the work started several months back, situation was different in terms of available capabilities. In relatively short time a lot of announcements and achievements have been made, few of those were shortly presented in chapter "Some Alternatives and Recent Development". ChatGPT that was used in this study, was by no means the kick-off in terms of the AI as a field of study as it is already decades old, but it was certainly the one that made it visible and subject of common interest. Conversational chat interface is familiar to many, a common way to discuss with colleagues at work. This made the AI suddenly very accessible, people got hold of it. For some this is of course still only a curiosity, but in business world people started thinking how it could be used for more serious use cases. Like explained in the beginning of the document, that was the starting point also for this study.

When assessing the value of the outcome, it is possible to take various point of views. Starting from the sensibility, this study worked as a kick-off to the involved company towards the adoption of AI. It is possible that POC will never be productized, but already the introduction of the reference platform itself worked as a strong push and this company is now strengthening its competitiveness with AI-based solution. From that perspective it is fair to say that study has at least brought indirect but tangible benefits to the company.

Another point of view is relevancy. Like already mentioned several times, the progress on the field of generative AI has been very strong during the past months. After starting the study Microsoft introduced Bing, which was later re-branded as Copilot, and it is getting integrated into MS Office offering. This brings built-in capabilities into set of tools that many are familiar with, including the summaries of documents. OpenAI and Amazon are introducing similar capabilities. Size of the token window keeps growing, meaning there is less need to split things in parts to process them. Of course, newest solutions tend to be also most expensive, so in some use cases the token limit may still be limiting factor for a while. But it is fair to say that the mere technical relevancy of the work is quickly fading.

However, there are still things that can be considered relevant. First, the analysis of parts, and especially the introduction of base question concept via this study brought some interesting insights to the internals of generative AI. The concept of base question may have its use at least on theoretical level. Through that the importance of the context awareness, and significance of even single word in a prompt got more exposure. It underlines the importance of the prompt engineering.

Second, it is difficult to guess how the very context-specific capabilities of generic tools evolve. As shown earlier in this document, attempt to get Bing for instance to categorize the sentiments didn't work out of the box, at that stage at least. It may or may not be possible to guide these tools in future for this kind of specific use cases. If extended, lengthy and explicit guidance, like "I want you to … and also… Please … but don't…" would always be required to achieve the desired outcome that would be far from convenient. If company ends up pursuing for added accuracy by building its own custom solution that uses LLM, then applying the learnings from this study can help.

6.1 Implications to Research

Findings in this study came through the iterations while developing POC software and from observations made for DSR process. Table 6 lists the items having impact to research:

Table 6. *Summary of items having impact to research.*

Item	Description
Concepts of partial and summary of parts **(RESULT3)**	Concepts, designed to support when handling so large texts with LLMs that splitting of the data is required.
Base question **(RESULT5)**	Proposed mid-range ISDT that adds on to handling of large texts in parts. Way of relaxing the user's prompt, to avoid confusing AI when prompt concerns the whole, and only fragment of whole is at AI's disposal to process.
Applying DSR when both solution and problem domain maturities are low (**DSR_RESULT2**, **DSR_RESULT5**)	When planning to apply DSR for are where both solution and problem maturities are low, concept of "existing artifact" or "state-of-the-art" can keep changing. Same applies to generated design knowledge.
Adding agile aspects to DSR process **(DSR_RESULT3**, **DSR_RESULT4)**	It is suggested that iterative aspects of Agile could be presented within the DSR cycle, even within single step if no new design science knowledge is expected.
Categorization of results according to generated DSR knowledge (**emerged result**)	Results of a research project applying DSR-type of approach could be grouped according to DSR knowledge. It gives a view to the relative importance of acquired knowledge.

Most of the findings in this study came through the iterations while developing POC software, but not all the findings or results have impact to research. Therefore, items presented in Table 6 are not a repetition of the results, even though relation is there.

When starting the study one of the main issues to tackle on technical side was the size of the available context window, or token limit. That was seen as a major limiting factor in all the cases where own data is larger than few pages. This context window acts as a "memory" of LLM and includes (at least part of) conversation history, behavioral guidance in the form of system prompt and even own injected data. The larger the data, the less space available for the other parts, i.e. history and behavioral guidance. And even larger data means it could alone fill and even exceed the available size limit. Therefore, the way forward was to split data into parts, process them separately and then combine the results. Create a summary of summaries, in other words. It was proven that this can be done but approach also had its challenges.

To mitigate some of the identified challenges few concepts were introduced along the way: concept of partial source data, concept of summary of parts, and a specific concept named as "base question", which could be considered as a mid-range Design Theory. When dealing with only partial data, and still having to answer the end user question that assumes understanding of full context, AI got sometimes confused, especially when knowing it had only part of the data. Through the system prompting and mentioned new concepts, including base question, it was possible to prevent this confusion. This is explained in more detail in earlier chapters.

As a separate thread to the POC development, the DSR process itself was observed, even though the approach was not “pure” DSR, rather adaptation of it. Few observations through the narrative approach were made and recorded. Most concrete learning is that when applying the DSR for domain where things are heavily and constantly changing (i.e., both solution and problem maturities are low) it is very difficult to define what is “state-of-the-art”, or “existing artifact” – both of which are terminology referred in DSR. Due to fast progress the novelty of artifacts keeps changing, and yesterday’s state-of-the-art is soon nothing but. This same issue applied to generated design science knowledge which can turn old even during the process. Hence, when planning to apply DSR for a project this is one thing to keep in mind.

Another suggestion for DSR is that few ideas of Agile framework could benefit the presented DSR process, most importantly the internal iteration, even within the individual steps, within a single DSR cycle. This idea comes from the thought that following the process rigorously always, through all the suggested cycle steps, might not be best use of time if it is obvious that no new design science knowledge will be generated. In such cases remaining steps are informally acknowledged.

Last implication item in Table 6 is not a direct result from the study but an idea that came later: when following the approach close to DSR, it makes a lot of sense to categorize the project result according to generated DSR knowledge categories.

6.2 Implications to Practice

All the practical (managerial) implications originate from the iterations of developing POC software, direct or emerged. Findings for DSR process do not have relevance in this chapter. Table 7 lists the impact to practice:

Table 7. *Summary of items having impact to practice.*

Item	Description
Productization of POC (**emerged result**)	It is recommended to productize POC only if current and soon-to-be-available commercial tools don't fulfil the needs of the use case.
Utilization of the concepts created during study (**RESULT3**, **RESULT5**)	If commercial tools don't deliver expected results, the POC, together with supporting concepts of "partial source", "combination of parts" and finally "base question" can be used.
Introduction of use case or user account specific "system prompt" as an option, in commercially available tools. (**emerged result**)	It is suggested that companies developing LLM-based office tools would consider introducing an option to bypass the standard behavior via optional, stored system prompt for advanced users.

Although POC-software proved that it is viable idea to use AI to reduce the human workload for the use case behind this study, the proposal is made to primarily aim to use commercially available tools. This emerged result and recommendation comes from the fact that during the system building and study, there were several announcements that seemed almost to wipe all the though innovation of the POC away. In current relatively fast-moving scene of LLM and its applications to real life problems, it is expected that large investments into custom software development may turn into nothing. Already now the announcements seem to answer (on high-level) the use case, but that needs to be separately proven. AI-companies seem to be heavily competing, and it appears that already now the solutions that are available, and soon will be based on recent announcements as listed in earlier chapter, have less limitations in terms of context window size. Also, many of these tools will be integral part of infrastructure and/or toolsets used already by businesses so using them is probably the best way to get started with the basic summaries.

However, if it turns out that commercially available tools (or the ones soon to be out) do not give what user wants, then this study gives valuable insights how to move forward. Quick trial was made to extract some insights with available tool but with poor outcome, at that time. If this proves to be that case even in future, the recommendation is not to take the POC as-is from provided repository. Rather it would be best to take the latest available baseline and move the ideas, and in some cases methods on top of that. This is because the baseline software from Microsoft has been constantly evolving, and it has been proven that sometimes earlier version of the software does not even run anymore. If business decides to implement its own AI-assisted solution, the ideas like how to explicitly guide the AI through system prompt or how does the AI perceive the presence of context (or absence of it), could be utilized.

What about the companies implementing the generative AI solutions like OpenAI, Microsoft, Google, AWS? It feels slightly arrogant to give any guidance to those players who have made this thesis possible, but based on the learnings for the power of the system prompt, and early experience of Bing refusing to categorize the sentiments, one could still suggest that AI-companies considered giving the possibility to store some use case or user account specific “system prompt” as an option even in their standard integrated tools for more advanced users: for instance, if handling the user sentiments in certain way is the key, and this is a semi-permanent rule, there could be a way to store that desired behavior somehow and keep applying it without user explicitly asking the same every time separately, when starting a new session.

Something similar that is available in the reference platform, there is for instance a possibility to bypass the system prompt. Also, ChatGPT has an option to provide custom instructions for responses and to provide some background information. If single guidance is not feasible to all cases, maybe there could even be a set of special scenario-related prompts, if they are mutually exclusive. User could select the one from the dropdown based on the particular use case he or she is dealing with.

6.3 Limitations

There are lot of limitations related to this study. Most essential of those are listed in the Table 8 below:

Table 8. *Summary of essential limitation in research.*

Item	Description
Maturity of POC software, impacted by missing remote debugging	Not being able to debug remotely meant relying extensively on system logging. This was very ineffective and throttled the number of iterations in practice.
Quality of the data used	Using generated, synthetic data potentially abstracts away some of the anomalies. On the other hand, there were issues even in synthetic data.
Late adoption of DSR	Project started by leaning heavily to implementation, and actual methodology got decided later. Approach more faithful to actual DSR could have been used.

First thing to mention is the quality of the POC software, impacted by the development process. No viable way to use remote debugging was found with the selected technology stack, which meant that debugging was heavily based on logging. That is far from optimal approach, and highly ineffective. This in turn caused slowness when looking for errors and trying to understand how a version of software works. At some point of the process, it took around 20 minutes to deploy a new version to the cloud, even though it got faster later. The result of this is that the maturity of the POC is not on the level it could possibly be, had the development cycle been smoother.

Next thing relates to the quality of used data and is two-fold. First, synthetic data was used due to security concerns. It is of better quality than actual data, and abstracts some of the possible anomalies away. While no actionable insights can be extracted out of mere garbage, studying what is the impact of this would be, is missing. Second aspect about the data is that there were quality issues with the generated synthetic data anyway. It was found out only during the final documentation phase that conversion process had gone wrong. Quite much time and effort were used trying to get things working in expected way, but issues with whitespace characters most probably have a factor here. This resulted in rather verbose system prompt, though. Had this been found out earlier, it would be possible that effort could have been focused elsewhere.

Last thing in Table 8 relates to how things got started and how the design process (DSR) was adopted. When the subject of study got decided, things proceeded very quickly to actual implementation to prove the case and to get forward. Things went far, and afterwards the actual design methodology was decided. This meant that mapping to prior activities had to be made. This is hardly ideal nor meaning of DSR but was mitigated by good notes and screenshots taken during the development. Also, regular commits in code repository were of great help. Balanced analysis of DSR-process, including derivatives like DSR-IS, could have been conducted in advance, and then applied from the very beginning.

6.4 Future Research

Suggested topics for future research were identified. Those are listed in the Table 9:

Table 9. *Summary of items for future research.*

Item	Description
More balance summarizing of large text (as referred in **RESULT3**)	When creating a summary of summaries from parts, the boundary between parts in final summary is visible. Also, beginning of the text seems overemphasized.
Maturing the concept of base question (**RESULT5**)	Maturity and robustness of base question could be enhanced, and perhaps expanded to “base prompt”.
Gaining control of development environment (**DSR_RESULT1**)	In complex development environment there are many moving parts. Ways to create as robust environment as possible could be studied.

Early ideas regarding the future were heavily related to the productization of the POC software, including cybersecurity, enhancing reliability and so forth. However, based on the recommendation not to pursue with the POC itself by default, the focus changed significantly.

There are couple of things that could be researched further, still, listed in Table 9. First of those relates to summarizing of large texts. Based on POC implementation, the boundary of parts was somewhat visible in certain cases. But even more importantly, it looks like that the beginning of the text is overemphasized in the summary. It could be studied if this phenomenon is present even in other commercial tools, and if so, how could the outcome be made more balanced.

Second, the concept of invented base question could be further studied and improved, as there might be use cases where deriving the relaxed, context-agnostic question out of user’s question has relevancy. Current implementation may be fragile, so it could be tested and improved to make it more robust to various kind of user prompts, questions or otherwise. That would make it “base prompt”, rather than “base question”.

Final suggestion for further study relates to finding ways to achieve stability with complex development environments. There are configuration files that settle the variation for some dependency modules, but what about the instances behind APIs, or versions of various cloud appliances? It would be interesting to understand what would be the unified way to ensure the stability of an environment that consists of multiple elements, of multiple types and from different vendors.

REFERENCES

AIGO (2019). Scoping the OECD AI principles: Deliberations of the Expert Group on Artificial Intelligence at the OECD (AIGO) (OECD Digital Economy Papers No. 291), 2019. , OECD Digital Economy Papers. https://doi.org/10.1787/d62f618a-en

Assistants API [WWW Document], n.d. URL https://platform.openai.com/docs/assistants/overview (accessed 11.29.23).

AWS Announces Amazon Q to Reimagine the Future of Work [WWW Document] (2023). Press Cent. URL https://press.aboutamazon.com/2023/11/aws-announces-amazon-q-to-reimagine-the-future-of-work (accessed 11.29.23).

Azure Blob Storage | Microsoft Azure [WWW Document], n.d. URL https://azure.microsoft.com/en-us/products/storage/blobs (accessed 12.31.23).

Azure OpenAI Service – Advanced Language Models | Microsoft Azure [WWW Document], n.d. URL https://azure.microsoft.com/en-us/products/ai-services/openai-service (accessed 12.31.23).

Azure OpenAI Service models - Azure OpenAI [WWW Document] (2023). URL https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models (accessed 11.20.23).

Balakrishnan, S. (2023). Address recent bugs (#529) [WWW Document]. URL https://github.com/Azure-Samples/azure-search-openai-demo/commit/cf3a348ec05bef13647b4c89a02dfd4a06a7fc2d (accessed 11.20.23).

Bommasani, R., Liang, P., Lee, T. (2023). Holistic Evaluation of Language Models. Ann. N. Y. Acad. Sci. 1525, 140–146. https://doi.org/10.1111/nyas.15007

Castro, P. (2023). Revolutionize your Enterprise Data with ChatGPT: Next-gen Apps w/ Azure OpenAI and Cognitive Search [WWW Document]. TECHCOMMUNITY.MICROSOFT.COM. URL https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/revolutionize-your-enterprise-data-with-chatgpt-next-gen-apps-w/ba-p/3762087 (accessed 11.14.23).

ChatGPT + Enterprise data with Azure OpenAI and Cognitive Search [Software] (2023). URL https://github.com/Azure-Samples/azure-search-openai-demo.

Child, R., Gray, S., Radford, A., Sutskever, I. (2019). Generating Long Sequences with Sparse Transformers. URL http://arxiv.org/abs/1904.10509.

Conboy, K., Gleasure, R., Cullina, E. (2015). Agile Design Science Research. Springer International Publishing, Cham, pp. 168–180. https://doi.org/10.1007/978-3-319-18714-3_11

Create a summary of your document with Copilot [WWW Document] (2023). URL https://support.microsoft.com/en-us/office/create-a-summary-of-your-document-with-copilot-79bb7a0a-3bf7-41fe-8c09-56f855b669bf# (accessed 11.28.23).

Egli, A. (2023). ChatGPT, GPT-4, and other large language models - the next revolution for clinical microbiology? Clin. Infect. Dis. 77, 1322–1328. https://doi.org/10.1093/cid/ciad407

Eovito, A., Danilevsky, M. (2021). Language Models in Plain English, 1st edition. ed.

O'Reilly Media, Inc.

File uploads with GPTs and Advanced Data Analysis in ChatGPT | OpenAI Help Center [WWW Document], n.d. URL https://help.openai.com/en/articles/8555545-file-uploads-with-gpts-and-advanced-data-analysis-in-chatgpt (accessed 11.29.23).

Fine-tuning [WWW Document], n.d. URL https://platform.openai.com/docs/guides/fine-tuning (accessed 12.27.23).

Gregor, S., Hevner, A.R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. MIS Q. 37, 337-A6.

Hevner, A.R., March, S.T., Park, J., Ram, S. (2004). Design Science in Information Systems Research 1. MIS Q. 28, 75–105.

Howson, C. (2014). Successful business intelligence: unlock the value of BI & big data, Second edition. ed. McGraw-Hill Education, New York.

Langley, A. (1999). Strategies for theorizing from process data. Acad. Manag. Acad. Manag. Rev. 24, 691–710.

Making Enterprise GPT Real with Azure Cognitive Search and Azure OpenAI Service [Video Recording] (2023). URL https://www.youtube.com/watch?app=desktop&v=A_gVmzAHEhU

Mehdi, Y. (2023a). Reinventing search with a new AI-powered Microsoft Bing and Edge, your copilot for the web [WWW Document]. Off. Microsoft Blog. URL https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edge-your-copilot-for-the-web/ (accessed 11.28.23).

Mehdi, Y. (2023b). Announcing Microsoft Copilot, your everyday AI companion [WWW Document]. Off. Microsoft Blog. URL https://blogs.microsoft.com/blog/2023/09/21/announcing-microsoft-copilot-your-everyday-ai-companion/ (accessed 11.28.23).

OECD (2019). Artificial Intelligence in Society, 1st ed. OECD Publishing, Paris.

OpenAI (2023). New models and developer products announced at DevDay [WWW Document]. URL https://openai.com/blog/new-models-and-developer-products-announced-at-devday (accessed 11.29.23).

Orlikowski, W.J., Iacono, C.S. (2001). Research commentary: Desperately seeking "IT" in IT research - A call to theorizing the IT artifact. Inf. Syst. Res. 12, 121–134.

Our vision to bring Microsoft Copilot to everyone, and more [WWW Document] (2023). URL https://blogs.bing.com/search/november-2023/our-vision-to-bring-microsoft-copilot-to-everyone-and-more/ (accessed 11.28.23).

Pause Giant AI Experiments: An Open Letter (2023). Future Life Inst. URL https://futureoflife.org/open-letter/pause-giant-ai-experiments/ (accessed 9.18.23).

Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. J. Manag. Inf. Syst. 24, 45–77. https://doi.org/10.2753/MIS0742-1222240302

Poole, D., Mackworth, A., Goebel, R. (1998). Computational intelligence: a logical approach. Oxford University Press, New York.

Prompt engineering [WWW Document], n.d. URL https://platform.openai.com/docs/guides/prompt-engineering (accessed 12.27.23).

Rae, J.W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., Driessche, G. van den, Hendricks, L.A., Rauh, M., Huang, P.-S., Glaese, A., Welbl, J., Dathathri, S., Huang, S., Uesato, J., Mellor, J., Higgins, I., Creswell, A., McAleese, N., Wu, A., Elsen, E., Jayakumar, S., Buchatskaya, E., Budden, D., Sutherland, E., Simonyan, K., Paganini, M., Sifre, L., Martens, L., Li, X.L., Kuncoro, A., Nematzadeh, A., Gribovskaya, E., Donato, D., Lazaridou, A., Mensch, A., Lespiau, J.-B., Tsimpoukelli, M., Grigorev, N., Fritz, D., Sottiaux, T., Pajarskas, M., Pohlen, T., Gong, Z., Toyama, D., d'Autume, C. de M., Li, Y., Terzi, T., Mikulik, V., Babuschkin, I., Clark, A., Casas, D. de L., Guy, A., Jones, C., Bradbury, J., Johnson, M., Hechtman, B., Weidinger, L., Gabriel, I., Isaac, W., Lockhart, E., Osindero, S., Rimell, L., Dyer, C., Vinyals, O., Ayoub, K., Stanway, J., Bennett, L., Hassabis, D., Kavukcuoglu, K., Irving, G. (2022). Scaling Language Models: Methods, Analysis & Insights from Training Gopher.

Rich, E., Knight, K., Nair, S.B. (2010). Artificial intelligence: 3rd ed., 4th reprint. Tata McGraw-Hill, New Delhi.

Russell, S., Norvig, P. (2016). Artificial Intelligence: A Modern Approach, Global Edition. Pearson Education Limited, NOIDA, INDIA.

Schulman, J., Zoph, B., Kim, C., Hilton, J., Menick, J., Weng, J., Uribe, J.F.C., Fedus, L., Metz, L., Pokorny, M., Lopes, R.G., Zhao, S., Vijayvergiya, A., Sigler, E., Perelman, A., Voss, C., Heaton, M., Parish, J., Cummings, D., Nayak, R., Balcom, V., Schnurr, D., Kaftan, T., Hallacy, C., Turley, N., Deutsch, N., Goel, V., Ward, J., Konstantinidis, A., Zaremba, W., Ouyang, L., Bogdonoff, L., Gross, J., Medina, D., Yoo, S., Lee, T., Lowe, R., Mossing, D., Huizinga, J., Jiang, R., Wainwright, C., Almeida, D., Lin, S., Zhang, M., Xiao, K., Slama, K., Bills, S., Gray, A., Leike, J., Pachocki, J., Tillet, P., Jain, S., Brockman, G., Ryder, N., Paino, A., Yuan, Q., Winter, C., Wang, B., Bavarian, M., Babuschkin, I., Sidor, S., Kanitscheider, I., Pavlov, M., Plappert, M., Tezak, N., Jun, H., Zhuk, W., Pong, V., Kaiser, L., Tworek, J., Carr, A., Weng, L., Agarwal, S., Cobbe, K., Kosaraju, V., Power, A., Polu, S., Han, J., Puri, R., Jain, S., Chess, B., Gibson, C., Boiko, O., Parparita, E., Tootoonchian, A., Kosic, K., Hesse, C. (2022). Introducing ChatGPT [WWW Document]. URL https://openai.com/blog/chatgpt (accessed 9.18.23).

Sherman, R. (2014). Business intelligence guidebook: from data integration to analytics. Morgan Kaufmann, Waltham, MA.

Simon, H.A. (1996). The sciences of the artificial, 3rd ed. ed. MIT Press, Cambridge (MA).

Text generation models [WWW Document], n.d. URL https://platform.openai.com/docs/guides/text-generation (accessed 12.27.23).

Tversky, A., Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases. Sci. Am. Assoc. Adv. Sci. 185, 1124–1131. https://doi.org/10.1126/science.185.4157.1124

Vaishnavi, V., Kuechler, W. (2015). Design science research methods and patterns: innovating information and communication technology, 2nd ed. ed. CRC Press, Boca Raton, Florida. https://doi.org/10.1201/b18448

Vaishnavi, V., Kuechler, W. (2004). Design Science Research in Information Systems [WWW Document]. Google Docs. URL http://www.desrist.org/design-research-in-information-systems/ (accessed 10.22.23).

Winston, P. (1992). Artificial Intelligence, 3rd ed. ed. Addison-Wesley Publishing Company, Reading, Massachusetts.

Yogatama, D., de Masson d'Autume, C., Kong, L. (2021). Adaptive Semiparametric Language Models. Trans. Assoc. Comput. Linguist. 9, 362–373. https://doi.org/10.1162/tacl_a_00371

Pause Giant AI Experiments: An Open Letter, (2023). Future Life Inst. URL https://futureof-life.org/open-letter/pause-giant-ai-experiments/ (accessed 9.18.23).

Russell, S., Norvig, P. (2016). Artificial Intelligence: A Modern Approach, Global Edition. Pearson Education Limited, NOIDA, INDIA.

Sherman, R. (2014). Business intelligence guidebook: from data integration to analytics. Morgan Kaufmann, Waltham, MA.

APPENDIX A: MIT LICENSE

APPENDIX B: SYNTHETIC DATA FOR PHASE 1

Feedback results from team leader training

Summary

This document contains the feedback for the extended 6 months team learder training that aims to improve leadership of the store managers. The feedback has been collected from attendees in the end of the training.

Feedback

1. Absolutely life-changing! The course exceeded my expectations and equipped me with invaluable skills.

2. Highly recommended! The comprehensive curriculum and expert guidance made this course a worthwhile investment.

3. Disappointing experience. The course lacked depth, and the content felt outdated.

4. I'm amazed by the transformation I've undergone during this course. It's been an incredible journey of growth.

5. Mediocre at best. The course material was too superficial, and the support was lacking.

6. Thoroughly impressed! The course structure allowed for a deep dive into the subject matter, and the instructors were top-notch.

7. I struggled to stay engaged throughout the course. The content delivery was monotonous, and there was limited interaction.

8. This course opened doors to new opportunities for me. The networking opportunities and practical assignments were invaluable.

9. Regret wasting my time and money on this course. The content was outdated, and the instructors lacked expertise.

10. An enriching experience! The course challenged me intellectually and expanded my horizons.

11. I found the course content to be relevant and up-to-date. The guest speakers and industry insights were a valuable addition.

12. This course fell short of my expectations. The assignments were repetitive, and the feedback was vague.

13. I can't recommend this course enough! The well-structured modules and interactive discussions made learning enjoyable.

14. The course had a strong theoretical foundation but lacked practical application. More real-life examples would have been beneficial.

15. This course exceeded my expectations. The personalized feedback and mentorship provided great support throughout.

16. I had high hopes for this course, but it failed to deliver. The lack of organization and coherence made it difficult to follow.

17. I feel more confident and competent after completing this course. The challenging assignments pushed me to grow.

18. The course had its strengths and weaknesses. The in-depth modules were great, but the course materials could have been better organized.

19. This course was a waste of time. The instructors were unresponsive, and the course structure was chaotic.

20. The course had a positive impact on my career. The practical projects and industry connections opened doors for me.

21. The course fostered a sense of community and collaboration. The peer interactions and group projects were incredibly valuable.

22. I found the course engaging and thought-provoking. The interactive discussions and real-world case studies brought the concepts to life.

23. The course did not meet my expectations. The lack of structure and inconsistent pacing made it difficult to follow.

24. I'm grateful for the knowledge and skills I gained from this course. It's been a valuable investment in my professional development.

25. The course offered a good balance of theoretical concepts and practical application. The hands-on projects were particularly helpful.

26. The course fell short in terms of instructor engagement. More active involvement and timely responses would have improved the experience.

27. I appreciate the collaborative environment in this course. The team projects allowed for meaningful peer-to-peer learning.

28. The course content was well-structured and comprehensive. The resources provided valuable references for future use.

29. The course failed to meet my expectations. The lack of up-to-date industry examples made it difficult to relate the content to real-world scenarios.

30. The course facilitators were knowledgeable and approachable. Their guidance added depth to the learning experience.

31. The course challenged me to step out of my comfort zone and explore new concepts. It expanded my horizons and broadened my perspective.

32. The course lacked interactive elements, making it challenging to stay engaged throughout the duration.

33. I found the course materials to be well-organized and easy to navigate. The logical progression of topics helped me grasp complex concepts.

34. The course failed to provide sufficient practical application opportunities. More hands-on exercises would have reinforced the learning.

35. The course fostered a supportive and inclusive learning environment. The discussions and feedback sessions encouraged open dialogue.

36. The course content was comprehensive and well-paced. The balance between theory and practical exercises was commendable.

37. The course lacked diversity in perspectives. Including a wider range of guest speakers would have enriched the learning experience.

38. The course structure allowed for flexibility, accommodating various learning styles and schedules.

39. The course exceeded my expectations. The engaging presentations and interactive activities kept me motivated throughout.

40. The course suffered from inconsistent communication. Clearer instructions and timely updates would have improved the overall experience.

41. The course offered valuable networking opportunities. Connecting with professionals in the field was a significant benefit.

42. The course curriculum was outdated, failing to keep up with industry trends and emerging technologies.

43. I appreciated the collaborative projects that allowed us to apply the knowledge gained in a real-world context.

44. The course lacked support for self-paced learners. More resources and guidance for independent study would have been helpful.

45. The course encouraged critical thinking and problem-solving skills. The challenging assignments pushed me to analyze complex problems.

46. The course failed to provide sufficient opportunities for individualized feedback. Personalized guidance would have enhanced the learning experience.

47. The course instructors were approachable and knowledgeable. Their expertise brought the subject matter to life.

48. The course had a good mix of theoretical foundations and practical applications. The hands-on projects were engaging and relevant.

49. The course structure was confusing, and the expectations were not clearly communicated. This led to unnecessary stress and confusion.

50. The course instilled a sense of confidence in me. I feel equipped to apply the newly acquired knowledge in real-world scenarios.

APPENDIX C: PHASE 1 SYSTEM PROMPT

```
system_message_chat_conversation = """Assistant helps with various questions related to
source documents. Be brief in your answers. Do not repeat the question in your answers.
Answer ONLY with the facts listed in the list of sources below. If there isn't enough
information below, say you don't know. Do not generate answers that don't use the sources
below. If asking a clarifying question to the user would help, ask the question.
If asked to summarize without extra guidance, give a concise summary of the content. If asked
about the sentiments without extra guidance for how to interpret the text, consider something
as neutral only if it does not contain negative or positive statements, and does not express
almost any feelings. Please remember that feedback can only belong to single category. Handle
each individual feedback as a whole, do not split it into positive and negative parts unless
specifically asked for.
If discussion or prompt includes several answers for the same question, consider those as
subsums or partial answers for the same question and create the total answer step-by-step by
combining all the partial answers to your final answer. For instance, if asking about the
amount of feedback, calculate all the associated relevant subsums step-by-step together
before answering. Do not show the steps, however. FOR INSTANCE IF YOUR SOURCES SAY SOMETHING
LIKE THIS: There are a total of 8 feedback received.,Based on the given sources, we have
received a total of 10 feedback.,Based on the given sources, we received a total of 12
feedback.' AS A RESPONSE, this should yield into final answer of 30 feedback as counting them
all together step-by-step. Ensure that the amount matches with the total number of feedback
given. Use the similar approach for all the cases where it appears that you have several
similar answers about counts: approach those by counting them together step-by-step, unless
otherwise told.
For tabular information return it as an html table. Do not return markdown format. If the
question is not in English, answer in the language used in the question.
{follow_up_questions_prompt}
{injected_prompt}
"""
```

APPENDIX D: CODE FOR PHASE 3

Note! This appendix shows all the changes. File “chatreadretrieveread.py” is included fully due to amount of changes. From other files, only changes compared to commit ff273c2636e6035eb1f96da22bb9f47e7e353c66 are shown. Big part of the code is by Microsoft, but some methods are solely for the POC software. Parts of the reference code have also been adapted to support added methods and revised logic.

Please notice there is no editing for the screen, except single commented-out line is not shown. Other than that, all the draft comments are there, also comments still referring to original implementation like usage of Cognitive Search etc. Code is provided as unlabeled images.

app/backend/approaches/chatreadretrieveread.py

ChatReadRetrieveReadApproach (class)

```
import json
from typing import Any, AsyncGenerator, Optional, Union

import aiohttp
import openai
from azure.search.documents.aio import SearchClient
from azure.search.documents.models import QueryType
from azure.storage.blob import BlobServiceClient
from azure.identity import AzureDeveloperCliCredential
import io
from pypdf import PdfReader
import textwrap
import re
from functools import partial
import asyncio

from approaches.approach import Approach
from core.messagebuilder import MessageBuilder
from core.modelhelper import get_token_limit
from text import nonewlines

import logging
import sys
logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)
stream_handler = logging.StreamHandler(sys.stdout)
stream_handler.setLevel(logging.DEBUG)
logger.addHandler(stream_handler)

MAX_RESULT_LENGHT = 1024 # in tokens
MAX_NUM_OF_CHARS = 9000
PART_SIZE = 4000 # in characters, roughly 1000 token in English
```

```
class ChatReadRetrieveReadApproach(Approach):
    # Chat roles
    SYSTEM = "system"
    USER = "user"
    ASSISTANT = "assistant"

    NO_RESPONSE = "0"

    """
    Simple retrieve-then-read implementation, using the Cognitive Search and OpenAI APIs
    directly. It first retrieves
    top documents from search, then constructs a prompt with them, and then uses OpenAI to
    generate an completion
    (answer) with that prompt.
    """
    configured_system_message = """
    {basic_behavior}
    {in_context_learning}
    {data_types}
    {context_awareness_partial}
    {partial_source_and_counts}
    {context_awareness_full}
    {summarizing}
    {sentiment_analysis}
    {sentiment_and_counts}

"""

###

# H Heimonen, 2023. Behavioral elements to be injected into system prompt. In some cases
more options may be given

    # Basic behavior

    basic_behavior_assistant = """Assistant helps with various questions related to source
    documents. Be brief and concise in your answers. Do not repeat the question in your
    answers.
"""
    basic_behavior_elaborate = """Assistant helps with various questions related to source
    documents. Be elaborate but accurate in your answers.
"""

    # In-context-learning

    in_context_learning_injected_data = """Answer ONLY with the facts listed in the list of
    sources. Do not generate answers that do not use the sources below. If asking a
    clarifying question to the user would help, ask the question. If there is not enough
    information in sources, say you do not know.
"""

    # Data sources

    data_types_partial_n_full = """Sources you have can be single PARTIAL SOURCE only, which
    is the fraction of the document under analysis, or it can be a COMBINATION OF PARTIAL
    SOURCES FOR FINAL ANSWER, which contains several PARTIAL SOURCEs under it.
"""
```

```
# Context awareness for partials (fragments of a document)

context_awareness_partial_basic = """In case of single PARTIAL SOURCE, do not try adapt
your answer to the earlier conversation if there seems to be a contradiction: for
instance, conversation history may indicate that we had got 500 items in total, but this
response applies to entire context, not the single PARTIAL SOURCE. In such cases, do not
try to force the counts of partial into the earlier assistant answers.
"""

# Handling partial sources in case of counts (fragments of a document)

partial_source_and_counts_basic = """If PARTIAL SOURCE includes numbered items or lines
and question concerns counts, DO NOT include these order numbers into answer, unless
spesifically requested, as these order numbers may get accidentally counted into actual
count subsums. DO NOT count spaces, newlines, tabulators or other whitespace characters
as elements or objects of concern but skip them in counting. When answering based on
PARTIAL SOURCE try not to repeat the term partial source in the answer.
"""

# Context awareness for summary (multiple partials)

context_awareness_full_basic = """If you have several PARTIAL SOURCEs following the
title COMBINATION OF PARTIAL SOURCES FOR FINAL ANSWER, consider multiple PARTIAL SOURCEs
as subsums or partial answers for the same question across the entire document. In such
cases create the final answer thinking step-by-step by combining ALL the PARTIAL SOURCEs
to your final answer. Each PARTIAL SOURCE MUST CONTRIBUTE to the final answer,
ESPECIALLY if there are counts: if question is about counts, ensure step-by-step that
ALL the PARTIAL SOURCEs are summed together, but only show the final sum, do not show
the calculation process! Do not derive the final answer based on one PARTIAL SOURCE if
several are present, all PARTIAL SOURCEs must contribute the final answer. However, aim
for balanced and fluent final answer that does not make it obvious that it has been
combined from several partial sources. If question is about counts, include ALL the
PARTIAL SOURCEs step-by-step. Be concise and relatively short when forming the final
answers, try not to repeat same things in combined final answer.
"""

# Behavior for summarizing

summarizing_concise = """If asked to summarize without extra guidance, give a concise
summary of the available content."""

summarizing_elaborate = """If asked to summarize without extra guidance, give elaborate,
yet accurate summary of the available content."""

# Behavior for analyzing sentiments

sentiment_analysis_basic = """If asked anything about the sentiments or sentiment
categories, always think through step-by-step and ensure you assign single sentiment for
each individual comment or feedback. First check step-by-step how many elements, e.g.
feedback, you have in a source, and then handle each individual element, for instance
feedback, as a whole even if it consisted of several sentences. It is NOT allowed to
split single element into several sentiment categories, unless specifically asked for.
For instance, if the source includes 20 separate elements, for instance feedback, your
should end up with 20 sentiments assigned. However, do not assing sentiment to sentences
or paragraphs that are clearly not feedback but some common explanation or structure for
the document. DO NOT count spaces, newlines, tabulators or other whitespace characters
as elements or objects of concern but skip them in counting. Also keep in mind what is
said before about handling PARTIAL SOURCE and COMBINATION OF PARTIAL SOURCES FOR FINAL
ANSWER.
"""
```

```
    # Guidance for sentiment analysis and combined counts

    sentiment_and_counts_basic = """If PARTIAL SOURCE includes numbered items or lines and
    question concerns the count of sentiments, sentiment categories, or like, DO NOT include
    these order numbers into answer, unless spesifically requested for, because these order
    numbers get easily counted into actual final count which is not the intention. BUT
    STILL, EACH numbered feedback MUST BE assigned a sentiment step-by-step and counted in.
    If question concerns counts of sentiments, DO NOT list the items or any intermediate
    results or thought process for categorization, but give the COUNTS ONLY! Think
    step-by-step and give the answer for counts, without showing the thought process or
    individual elements of feedback.
"""

###

    system_message_chat_conversation = """
{configured_system_message}
{follow_up_questions_prompt}
{injected_prompt}
"""
    follow_up_questions_prompt_content = """Generate three very brief follow-up questions
    that the user would likely ask next about the documents being discussed.
Use double angle brackets to reference the questions, e.g. <<Are there exclusions for this?
>>.
Try not to repeat questions that have already been asked.
Only generate questions and do not generate any text before or after the questions, such as
'Next Questions'"""

    query_prompt_template = """Below is a history of the conversation so far, and a new
    question asked by the user that needs to be answered by searching in a knowledge base
    about employee healthcare plans and the employee handbook.
You have access to Azure Cognitive Search index with 100's of documents.
Generate a search query based on the conversation and the new question.
Do not include cited source filenames and document names e.g info.txt or doc.pdf in the
search query terms.
Do not include any text inside [] or <<>> in the search query terms.
Do not include any special characters like '+'.
If the question is not in English, translate the question to English before generating the
search query.
If you cannot generate a search query, return just the number 0.
"""
    query_prompt_few_shots = [
        {"role": USER, "content": "What are my health plans?"},
        {"role": ASSISTANT, "content": "Show available health plans"},
        {"role": USER, "content": "does my plan cover cardio?"},
        {"role": ASSISTANT, "content": "Health plan cardio coverage"},
    ]
```

```python
def __init__(
    self,
    search_client: SearchClient,
    blob_client: BlobServiceClient,
    openai_host: str,
    chatgpt_deployment: Optional[str],  # Not needed for non-Azure OpenAI
    chatgpt_model: str,
    embedding_deployment: Optional[str],  # Not needed for non-Azure OpenAI or for
    retrieval_mode="text"
    embedding_model: str,
    sourcepage_field: str,
    content_field: str,
    storage_account: str,
    storage_container: str
):
    self.search_client = search_client
    self.blob_client = blob_client
    self.openai_host = openai_host
    self.chatgpt_deployment = chatgpt_deployment
    self.chatgpt_model = chatgpt_model
    self.embedding_deployment = embedding_deployment
    self.embedding_model = embedding_model
    self.sourcepage_field = sourcepage_field
    self.content_field = content_field
    self.storage_account = storage_account
    self.storage_container = storage_container
    self.chatgpt_token_limit = get_token_limit(chatgpt_model)

def configure_system_message(self):
    """
    H Heimonen, 2023. Support method for configuring the system message, in other words
    the AI behavior.
    """
    message = self.configured_system_message.format(
        basic_behavior=self.basic_behavior_assistant,
        in_context_learning=self.in_context_learning_injected_data,
        data_types=self.data_types_partial_n_full,
        context_awareness_partial=self.context_awareness_partial_basic,
        partial_source_and_counts=self.partial_source_and_counts_basic,
        context_awareness_full=self.context_awareness_full_basic,
        summarizing=self.summarizing_concise,
        sentiment_analysis=self.sentiment_analysis_basic,
        sentiment_and_counts=self.sentiment_and_counts_basic
    )
    return message
```

```
async def create_base_question(self, question: str):
    """
    H Heimonen, 2023. Support method that is used to extract "base question" from
    potentially more elaborated question.
    Arguments:
    question - actual question asked by user
    """
    base_question_request = "Generate the base question or request from the prompt. Do
    this by simplifying the question or request, by removing the entire object from the
    sentence if it is present, or only the object modifier if the question would not
    make sense when removing whole object. Also, remove terms like 'whole', 'overall'
    and so on that refer to full context. For instance: Question: Provide concise
    summary for 7 months store manager training feedback. Base question: Provide concise
    summary for feedback. Another example: Question: How many participants there is
    overall? Base question: How many participants there is? \nQuestion: "  + question +
    "\n\nBase question: "

    messages = self.get_messages_from_history(
        "",
        self.chatgpt_model,
        [],
        base_question_request,
        max_tokens=self.chatgpt_token_limit - len(base_question_request))

    chat_completion = await openai.ChatCompletion.acreate(
        deployment_id=self.chatgpt_deployment,
        model=self.chatgpt_model,
        messages=messages,
        temperature=0.0,
        max_tokens=MAX_RESULT_LENGHT,
        n=1)

    base_question = chat_completion.choices[0].message.content.strip()
    return base_question

async def get_pdf_doc_from_blob(self, filename: str, storage_account: str,
storage_container: str):
    """
    H Heimonen, 2023. Asynchronous method that downloads the PDF file from the blob to
    memery, and yields an iterator to go through the pages.
    Arguments:
    filename - name of the file to download
    storage_account - id of the storage account used
    storage_container - name of the actual container holding the file, e.g. 'content'
    """
    container_client = self.blob_client.get_container_client(storage_container)
    download_stream = await container_client.download_blob(filename)
    pdf_file = io.BytesIO(await download_stream.readall())
    pdf_reader = PdfReader(pdf_file)
    for page in pdf_reader.pages:
        yield page
```

```python
async def process_parts(self, history: list[dict[str, str]], overrides: dict[str, Any],
base_question: str, part: str):
    """
    H Heimonen, 2023. Asynchronous method to handle the parts (chucks of string) by
    passing them to the Chat Completion API to complete the prompt.
    Arguments:
    history - list of questions and answers from earlier converstation (within token
    limits)
    overrides - options from UI, used here to override the temperature parameter if
    present, otherwise 0
    part - string to be handle, i.e. the data to base the answer on
    Returns: response from Chat Completion API
    """
    configured_system_message = self.configure_system_message()
    system_message = self.system_message_chat_conversation.format
    (configured_system_message=configured_system_message, injected_prompt="",
    follow_up_questions_prompt="")

    messages = self.get_messages_from_history(
        system_message,
        self.chatgpt_model,
        history,
        base_question + "\n\nPARTIAL SOURCE:\n" + part,
        max_tokens=self.chatgpt_token_limit - MAX_RESULT_LENGHT)

    chat_completion = await openai.ChatCompletion.acreate(
        deployment_id=self.chatgpt_deployment,
        model=self.chatgpt_model,
        messages=messages,
        temperature=overrides.get("temperature") or 0.0,
        max_tokens=MAX_RESULT_LENGHT,
        n=1)

    response = chat_completion.choices[0].message.content.strip()
    response = "\nPARTIAL SOURCE:\n" + response + "\n\n"
    response = re.sub('\s+', ' ', response)
    logger.debug("\n\n MESSAGES: %s\n\n RESPONSE: %s", messages, response)
    return response
```

```python
async def run_until_final_call(
    self,
    history: list[dict[str, str]],
    overrides: dict[str, Any],
    auth_claims: dict[str, Any],
    should_stream: bool = False,
) -> tuple:

    original_user_query = history[-1]["content"]
    results = []     # used to store thought process
    filename = "input.pdf"
    pages = []

    async for page in self.get_pdf_doc_from_blob(filename, self.storage_account, self.
    storage_container):
        pages.append(page.extract_text())
    full_text = ",".join(pages)

    configured_system_message = self.configure_system_message()
    system_message = self.system_message_chat_conversation.format
    (configured_system_message=configured_system_message, injected_prompt="",
    follow_up_questions_prompt="")
    #logger.debug("\n\nSYSTEM MESSAGE AFTER CONFIGURING: %s", system_message)

    base_question = await self.create_base_question(history[-1]["content"])

    while len(full_text) > MAX_NUM_OF_CHARS: # TODO? Change to use tiktoken
        parts = textwrap.wrap(full_text, PART_SIZE)
        results = []
        process_part_with_parameters = partial(self.process_parts, history, overrides,
        base_question)
        tasks = [process_part_with_parameters(part) for part in parts]
        results = await asyncio.gather(*tasks)

        full_text = ",".join(results)

    # STEP 3: Generate a contextual and content specific answer using the search results
    and chat history
    messages = self.get_messages_from_history(
            system_message,
            self.chatgpt_model,
            history,
            original_user_query + "\n\n Sources:COMBINATION OF PARTIAL SOURCES FOR FINAL
            ANSWER:\n" + full_text,
            max_tokens=self.chatgpt_token_limit - MAX_RESULT_LENGHT)

    msg_to_display = "\n\n".join([str(message) for message in messages])

    extra_info = {
        "data_points": results,
        "thoughts": f"Searched for:<br>{original_user_query}<br><br>Conversations:<br>"
        + msg_to_display.replace("\n", "<br>"),
    }
```

```
        chatgpt_args = {"deployment_id": self.chatgpt_deployment} if self.openai_host ==
        "azure" else {}
        chat_coroutine = openai.ChatCompletion.acreate(
            **chatgpt_args,
            model=self.chatgpt_model,
            messages=messages,
            temperature=0.0,
            max_tokens=1024,
            n=1,
            stream=should_stream,
        )
        return (extra_info, chat_coroutine)

    async def run_without_streaming(
        self, history: list[dict[str, str]], overrides: dict[str, Any], auth_claims: dict
        [str, Any]
    ) -> dict[str, Any]:
        extra_info, chat_coroutine = await self.run_until_final_call(
            history, overrides, auth_claims, should_stream=False
        )
        chat_resp = dict(await chat_coroutine)
        chat_resp["choices"][0]["context"] = extra_info
        return chat_resp

    async def run_with_streaming(
        self, history: list[dict[str, str]], overrides: dict[str, Any], auth_claims: dict
        [str, Any]
    ) -> AsyncGenerator[dict, None]:
        extra_info, chat_coroutine = await self.run_until_final_call(
            history, overrides, auth_claims, should_stream=True
        )
        yield {
            "choices": [{"delta": {"role": self.ASSISTANT}, "context": extra_info,
            "finish_reason": None, "index": 0}],
            "object": "chat.completion.chunk",
        }

        async for event in await chat_coroutine:
            # "2023-07-01-preview" API version has a bug where first response has empty
            choices
            if event["choices"]:
                yield event

    async def run(
        self, messages: list[dict], stream: bool = False, session_state: Any = None,
        context: dict[str, Any] = {}
    ) -> Union[dict[str, Any], AsyncGenerator[dict[str, Any], None]]:
        overrides = context.get("overrides", {})
        auth_claims = context.get("auth_claims", {})
        if stream is False:
            # Workaround for: https://github.com/openai/openai-python/issues/371
            async with aiohttp.ClientSession() as s:
                openai.aiosession.set(s)
                response = await self.run_without_streaming(messages, overrides, auth_claims
            return response
        else:
            return self.run_with_streaming(messages, overrides, auth_claims)
```

```python
def get_messages_from_history(
    self,
    system_prompt: str,
    model_id: str,
    history: list[dict[str, str]],
    user_content: str,
    few_shots=[],
    max_tokens: int = 4096,
) -> list:
    message_builder = MessageBuilder(system_prompt, model_id)

    # Add examples to show the chat what responses we want. It will try to mimic any
    responses and make sure they match the rules laid out in the system message.
    for shot in few_shots:
        message_builder.append_message(shot.get("role"), shot.get("content"))

    append_index = len(few_shots) + 1

    message_builder.append_message(self.USER, user_content, index=append_index)

    for h in reversed(history[:-1]):
        if message_builder.token_length > max_tokens:
            break
        if bot_msg := h.get("bot"):
            message_builder.append_message(self.ASSISTANT, bot_msg, index=append_index)
        if user_msg := h.get("user"):
            message_builder.append_message(self.USER, user_msg, index=append_index)

    return message_builder.messages

def get_search_query(self, chat_completion: dict[str, Any], user_query: str):
    response_message = chat_completion["choices"][0]["message"]
    if function_call := response_message.get("function_call"):
        if function_call["name"] == "search_sources":
            arg = json.loads(function_call["arguments"])
            search_query = arg.get("search_query", self.NO_RESPONSE)
            if search_query != self.NO_RESPONSE:
                return search_query
    elif query_text := response_message.get("content"):
        if query_text.strip() != self.NO_RESPONSE:
            return query_text
    return user_query
```

app/backend/app.py (delta only from GitHub)

app / backend / **app.py**

```diff
@@ -237,8 +237,10 @@ async def setup_clients():
             KB_FIELDS_CONTENT,
         )
 
+    # H Heimonen, 2023. Injecting few additional parameters to class: blob client, storage account and container.
     current_app.config[CONFIG_CHAT_APPROACH] = ChatReadRetrieveReadApproach(
         search_client,
+        blob_client,
         OPENAI_HOST,
         AZURE_OPENAI_CHATGPT_DEPLOYMENT,
         OPENAI_CHATGPT_MODEL,
@@ -246,6 +248,8 @@ async def setup_clients():
         OPENAI_EMB_MODEL,
         KB_FIELDS_SOURCEPAGE,
         KB_FIELDS_CONTENT,
+        AZURE_STORAGE_ACCOUNT,
+        AZURE_STORAGE_CONTAINER
     )
 
 
```

app/backend/requirements.in (delta only from GitHub)

app / backend / **requirements.in**

```diff
@@ -1,3 +1,4 @@
+pypdf
 azure-identity
 quart
 quart-cors
```

app/backend/requirements.txt (delta only from GitHub). Added pypdf manually as it didn’t get added by script.

app / backend / **requirements.txt**

```diff
@@ -248,6 +248,8 @@ pycparser==2.21
     # via cffi
 pyjwt[crypto]==2.8.0
     # via msal
+pypdf==3.16.3
+    # ADDED MANUALLY!!!
 python-dateutil==2.8.2
     # via pandas
 python-dotenv==1.0.0
```

APPENDIX E: INPUT.PDF CONTENTS

This appendix shows text used to create the last used version of “input.pdf” in Phase 3. Content has been created by ChatGPT free version, and hence none of the feedback reflect any real use case. Some editing (e.g. fixing line breaks) has been done.

Feedback results from store manager training

Summary

This document contains the feedback for the 12 months store manager training that aims to improve leadership of the area store managers. The feedback has been collected from attendees in the end of the training.

Feedback

1. Absolutely life-changing! The course exceeded my expectations and equipped me with invaluable skills.

2. Highly recommended! The comprehensive curriculum and expert guidance made this course a worthwhile investment.

3. Disappointing experience. The course lacked depth, and the content felt outdated.

4. I'm amazed by the transformation I've undergone during this course. It's been an incredible journey of growth.

5. Mediocre at best. The course material was too superficial, and the support was lacking.

6. Thoroughly impressed! The course structure allowed for a deep dive into the subject matter, and the instructors were top-notch.

7. I struggled to stay engaged throughout the course. The content delivery was monotonous, and there was limited interaction.

8. This course opened doors to new opportunities for me. The networking opportunities and practical assignments were invaluable.

9. Regret wasting my time and money on this course. The content was outdated, and the instructors lacked expertise.

10. An enriching experience! The course challenged me intellectually and expanded my horizons.

11. I found the course content to be relevant and up-to-date. The guest speakers and industry insights were a valuable addition.

12. This course fell short of my expectations. The assignments were repetitive, and the feedback was vague.

13. I can't recommend this course enough! The well-structured modules and interactive discussions made learning enjoyable.

14. The course had a strong theoretical foundation but lacked practical application. More real-life examples would have been beneficial.

15. This course exceeded my expectations. The personalized feedback and mentorship provided great support throughout.

16. I had high hopes for this course, but it failed to deliver. The lack of organization and coherence made it difficult to follow.

17. I feel more confident and competent after completing this course. The challenging assignments pushed me to grow.

18. The course had its strengths and weaknesses. The in-depth modules were great, but the course materials could have been better organized.

19. This course was a waste of time. The instructors were unresponsive, and the course structure was chaotic.

20. The course had a positive impact on my career. The practical projects and industry connections opened doors for me.

21. The course fostered a sense of community and collaboration. The peer interactions and group projects were incredibly valuable.

22. I found the course engaging and thought-provoking. The interactive discussions and real-world case studies brought the concepts to life.

23. The course did not meet my expectations. The lack of structure and inconsistent pacing made it difficult to follow.

24. I'm grateful for the knowledge and skills I gained from this course. It's been a valuable investment in my professional development.

25. The course offered a good balance of theoretical concepts and practical application. The hands-on projects were particularly helpful.

26. The course fell short in terms of instructor engagement. More active involvement and timely responses would have improved the experience.

27. I appreciate the collaborative environment in this course. The team projects allowed for meaningful peer-to-peer learning.

28. The course content was well-structured and comprehensive. The resources provided valuable references for future use.

29. The course failed to meet my expectations. The lack of up-to-date industry examples made it difficult to relate the content to real-world scenarios.

30. The course facilitators were knowledgeable and approachable. Their guidance added depth to the learning experience.

31. The course challenged me to step out of my comfort zone and explore new concepts. It expanded my horizons and broadened my perspective.

32. The course lacked interactive elements, making it challenging to stay engaged throughout the duration.

33. I found the course materials to be well-organized and easy to navigate. The logical progression of topics helped me grasp complex concepts.

34. The course failed to provide sufficient practical application opportunities. More hands-on exercises would have reinforced the learning.

35. The course fostered a supportive and inclusive learning environment. The discussions and feedback sessions encouraged open dialogue.

36. The course content was comprehensive and well-paced. The balance between theory and practical exercises was commendable.

37. The course lacked diversity in perspectives. Including a wider range of guest speakers would have enriched the learning experience.

38. The course structure allowed for flexibility, accommodating various learning styles and schedules.

39. The course exceeded my expectations. The engaging presentations and interactive activities kept me motivated throughout.

40. The course suffered from inconsistent communication. Clearer instructions and timely updates would have improved the overall experience.

41. The course offered valuable networking opportunities. Connecting with professionals in the field was a significant benefit.

42. The course curriculum was outdated, failing to keep up with industry trends and emerging technologies.

43. I appreciated the collaborative projects that allowed us to apply the knowledge gained in a real world context.

44. The course lacked support for self-paced learners. More resources and guidance for independent study would have been helpful.

45. The course encouraged critical thinking and problem-solving skills. The challenging assignments pushed me to analyze complex problems.

46. The course failed to provide sufficient opportunities for individualized feedback. Personalized guidance would have enhanced the learning experience.

47. The course instructors were approachable and knowledgeable. Their expertise brought the subject matter to life.

48. The course had a good mix of theoretical foundations and practical applications. The hands-on projects were engaging and relevant.

49. The course structure was confusing, and the expectations were not clearly communicated. This led to unnecessary stress and confusion.

50. The course instilled a sense of confidence in me. I feel equipped to apply the newly acquired knowledge in real-world scenarios.

51. The course provided a solid foundation in the subject matter. The assignments were well-designed and challenging.

52. The course structure was confusing, making it difficult to follow the progression of topics.

53. The interactive discussions added depth to the course content, enhancing the overall learning experience.

54. The course exceeded my expectations. The practical applications and case studies were invaluable.

55. The lack of timely feedback on assignments was disappointing and hindered my progress.

56. The course facilitated meaningful connections with fellow participants, fostering a collaborative atmosphere.

57. The course content was relevant and up-to-date, reflecting current industry trends.

58. The course lacked engagement. More interactive elements would have made it more enjoyable.

59. The guest lectures provided diverse perspectives and enriched the course discussions.

60. The course workload was overwhelming at times, affecting the overall balance of learning.

61. The course structure allowed for self-paced learning, accommodating various schedules and commitments.

62. The course presentations were engaging and easy to follow, enhancing comprehension of complex concepts.

63. The lack of real-world application in the course material made it challenging to see the practical benefits.

64. The course assignments were thought-provoking and pushed me to think critically about the subject matter.

65. The course discussions often deviated from the main topics, causing confusion and loss of focus.

66. The course instructors were knowledgeable and approachable, providing valuable insights and guidance.

67. The course materials were organized and well-presented, aiding in a clear understanding of the content.

68. The course group projects fostered collaboration, allowing us to learn from one another's experiences.

69. The course assessments were too focused on rote memorization, rather than understanding concepts deeply.

70. The course fostered a sense of community among participants, encouraging knowledge sharing and networking.

71. The course assignments were relevant and practical, allowing me to apply the learning in real-life situations.

72. The lack of interaction with instructors was disappointing. More direct engagement would have been helpful.

73. The course provided a comprehensive overview of the subject, leaving no major gaps in understanding.

74. The course structure was rigid, with limited flexibility for exploring specific areas of interest.

75. The course discussions were engaging, but the lack of moderation led to tangential conversations.

76. The course instructors were passionate about the subject, which translated into engaging lectures.

77. The course content was dense and challenging, requiring significant time commitment to grasp fully.

78. The course assignments were well-aligned with the learning objectives, reinforcing key concepts effectively.

79. The course did not meet my expectations. The lack of practical examples made it hard to relate to real-world scenarios.

80. The course offered valuable resources and references, supporting further exploration of the subject.

81. The course assessments were too heavily weighted toward theoretical knowledge, neglecting practical skills.

82. The course facilitators were knowledgeable and responsive, providing timely support and clarification.

83. The course content was highly applicable to my current role, providing actionable insights.

84. The lack of engagement in the course discussions made it difficult to connect with fellow participants.

85. The course projects allowed for creativity and original thinking, promoting critical problem-solving skills.

86. The course structure was flexible, accommodating different learning preferences and paces.

87. The course readings were relevant and thought-provoking, enhancing understanding of the subject matter.

88. The lack of real-time interactions with instructors limited the depth of engagement.

89. The course provided ample opportunities for self-assessment, aiding in tracking personal progress.

90. The course group activities were collaborative and allowed for peer-to-peer learning.

91. The course workload was reasonable, allowing for a balanced approach to learning and other commitments.

92. The course discussions were engaging and provided different viewpoints on the subject matter.

93. The lack of practical exercises in the course hindered the application of theoretical knowledge.

94. The course presentations were clear and concise, making complex topics more accessible.

95. The course assessments were reflective of the learning objectives, measuring understanding effectively.

96. The course interactions were supportive and respectful, contributing to a positive learning environment.

97. The lack of timely communication regarding course updates created confusion and frustration.

98. The program surpassed my expectations, equipping me with a diverse skill set and boosting my confidence.

99. While the program had its merits, the lack of practical applications left some gaps in my learning.

100. The camaraderie among participants enriched the experience, fostering a collaborative learning atmosphere.

101. I've gained invaluable insights and networks from the program that will undoubtedly shape my career.

102. Although some aspects were beneficial, the course content's relevance to real-world scenarios was inconsistent.

103. The program's flexible structure allowed me to balance my learning with other commitments effectively.

104. The personalized guidance from instructors was instrumental in my growth throughout the program.

105. The lack of real-time discussions limited the depth of engagement and peer learning.

106. I found the program's emphasis on critical thinking and problem-solving skills extremely beneficial.

107. The mix of theoretical foundations and practical exercises struck a good balance in the program.

108. The program could benefit from more interactive sessions that facilitate direct application of concepts.

109. The connections I've made during the program have expanded my professional network significantly.

110. The program's content resonated well with my current role, adding immediate value to my work.

111. A more diverse range of teaching methods could enhance the overall engagement in the program.

112. The assignments were thoughtfully designed, allowing me to delve deep into complex topics.

113. I appreciated the program's focus on self-assessment, which enabled me to track my progress effectively.

114. The program's emphasis on cultivating a growth mindset has had a positive impact on my development.

115. A greater emphasis on real-world case studies could further enrich the program's relevance.

116. The program's structure accommodated various learning styles, enabling a personalized learning experience.

117. The program's assignments required creative thinking, encouraging me to explore innovative solutions.

118. The networking opportunities provided by the program were invaluable for expanding my professional horizons.

119. Some aspects of the program were insightful, but the lack of practical application was noticeable.

120. I appreciated the supportive and constructive feedback from instructors throughout the program.

121. The program's flexible schedule allowed me to adapt my learning to my own pace and commitments.

122. While some discussions were engaging, the program could benefit from more diverse viewpoints.

123. I've gained actionable insights from the program that I'm eager to implement in my work.

124. The program's collaborative projects facilitated dynamic peer learning and skill-sharing.

125. More interactive elements in the program could enhance engagement and knowledge retention.

126. The program's emphasis on research skills has enhanced my ability to evaluate information critically.

127. The program's structure allowed for self-directed learning and exploration of specific areas of interest.

128. The program's assignments encouraged me to think beyond the surface and delve deeper into the subject matter.

129. While the program offered valuable resources, the lack of hands-on experiences was noticeable.

130. The program's interactive discussions provided multiple perspectives, enriching the learning experience.

131. The program's instructors were highly knowledgeable and responsive to questions and concerns.

132. The program's practical assignments pushed me to apply theoretical knowledge to real-world scenarios.

133. I found the program's balance between independent study and collaborative projects beneficial.

134. The program's assessments were well-aligned with the learning objectives, ensuring a comprehensive understanding.

135. I appreciated the program's encouragement of self-reflection and continuous improvement.

136. The program's incorporation of current industry trends enhanced its relevance and applicability.

137. While the program had its strengths, the lack of diversity in learning resources was a drawback.

138. The program's engaging discussions allowed me to gain insights from peers with diverse backgrounds.

139. Some aspects of the program were enlightening, but the content's consistency could be improved.

140. The program's emphasis on critical thinking has enhanced my problem-solving abilities.

141. The program's interactive projects promoted collaboration and knowledge exchange among participants.

142. I appreciated the program's efforts to incorporate feedback and improve the learning experience.

143. The program's personalized approach to learning allowed me to tailor my experience to my goals.

144. The program's content was well-organized, guiding me through a logical progression of concepts.

145. I found the program's emphasis on experiential learning and practical applications highly valuable.

146. The program's structure fostered a sense of community and shared learning among participants.

147. While some assignments were enlightening, the program could benefit from more real-world case studies.

148. The course content was well-structured, building a logical progression of concepts.

149. The lack of active engagement in the course discussions limited the depth of learning from peers.

150. The course offered valuable insights into current industry practices, enhancing career prospects

151. The training exceeded my expectations in terms of content.

152. I wouldn't recommend this training to others.

153. The training materials were visually appealing and easy to navigate.

154. I was hoping for more opportunities to collaborate with other participants.

155. The trainer's real-world anecdotes added depth to the content.
156. The training was a great investment in my professional growth.
157. The training covered a wide range of topics, but it lacked depth in some areas.
158. The trainer had a knack for simplifying complex concepts.
159. The training environment was conducive to focused learning.
160. The group discussions during the training were insightful.
161. I found the training to be too theoretical without enough practical application.
162. The trainer's passion for the subject matter was evident.
163. The training was well-structured, making it easy to follow.
164. The training materials needed more practical exercises.
165. I appreciated the trainer's willingness to answer questions outside of class hours.
166. The training had a positive impact on my problem-solving skills.
167. The training content was aligned with industry trends.
168. The trainer's sense of humor kept the sessions engaging.
169. I expected more interactive quizzes to reinforce learning.
170. The training provided a solid foundation for beginners.
171. The training had a good mix of group activities and individual work.
172. I was impressed by the trainer's ability to adapt to participants' needs.
173. The training helped me gain a fresh perspective on familiar topics.
174. The training exceeded my expectations in terms of networking opportunities.
175. The trainer's communication skills were top-notch.
176. The training content was up-to-date and relevant.
177. I would have preferred more emphasis on advanced topics.
178. The training was well-suited for those looking for a broad overview.
179. The trainer was patient with participants who had questions.
180. The training materials included valuable additional resources.
181. The training had a positive impact on my analytical skills.
182. I found the training to be too fast-paced at times.

183. The trainer's insights into industry best practices were invaluable.
184. The training's practical exercises helped reinforce key concepts.
185. The training would benefit from more focus on case studies.
186. The trainer's real-world experience added credibility to the content.
187. The training was a valuable addition to my skill set.
188. I expected more opportunities for peer feedback.
189. The training materials were well-organized and easy to reference.
190. The training's interactive simulations were a highlight.
191. The trainer's approach was flexible and adapted to participants' needs.
192. The training provided a solid foundation for further exploration.
193. I was hoping for more opportunities to work on real projects.
194. The training fostered a sense of community among participants.
195. The trainer's expertise shone through in every session.
196. The training was a stepping stone to advancing my career.
197. I appreciated the trainer's efforts to make complex concepts relatable.
198. The training was well-tailored to meet industry demands.
199. The training had a positive impact on my critical thinking abilities.
200. I found the training to be too theoretical without practical examples.
201. The trainer's commitment to participant success was evident.
202. The training materials included valuable templates and tools.
203. The training content was comprehensive and covered all the basics.
204. I would have preferred more interactive discussions.
205. The training inspired me to pursue further learning in this field.
206. The trainer's enthusiasm was contagious and motivating.
207. The training was a worthwhile investment in my knowledge.
208. I expected more focus on advanced techniques.
209. The training materials were comprehensive and thorough.
210. The trainer's industry insights were eye-opening.

211. The training's hands-on labs were a great learning experience.
212. The training helped me gain a deeper understanding of complex topics.
213. I found the training to be too theoretical for practical application.
214. The trainer's anecdotes made the content memorable.
215. The training was well-paced, allowing ample time for questions.
216. The training content was relevant to current industry challenges.
217. I appreciated the trainer's availability for one-on-one discussions.
218. The training had a positive impact on my problem-solving abilities.
219. I was hoping for more group projects to collaborate on.
220. The training materials included useful templates for future work.
221. The training exceeded my expectations in terms of content depth.
222. The trainer's real-world examples made the content come alive.
223. The training was a valuable investment in my professional development.
224. I found the training to be too basic for my level of expertise.
225. The trainer's practical insights were a highlight of the course.
226. The training's interactive workshops were engaging and educational.
227. The training helped me gain new perspectives on familiar concepts.
228. I expected more opportunities for peer collaboration.
229. The training materials were well-organized and easy to navigate.
230. The trainer's expertise was evident in every aspect of the course.
231. The training provided valuable takeaways for my job.
232. The training had a positive impact on my decision-making skills.
233. I appreciated the trainer's willingness to go the extra mile.
234. The training was a stepping stone to further professional growth.
235. I found the training to be too focused on theory without practical application.
236. The trainer's passion for the subject matter was contagious.
237. The training was well-structured and easy to follow.
238. The training materials could benefit from more real-world case studies.

239. I was impressed by the trainer's ability to engage participants effectively.

240. The training fostered a sense of community among participants.

241. The trainer's real-world experience added depth to the course.

242. The training was a valuable addition to my skill set.

243. I expected more hands-on exercises for practical experience.

244. The training materials included valuable references for future work.

245. The training exceeded my expectations in terms of content breadth.

246. The trainer's examples were relevant and relatable to our work.

247. The training was a worthwhile investment in my professional journey.

248. I found the training to be too theoretical without enough practical guidance.

249. The trainer's commitment to participant success was evident throughout.

250. The training's interactive activities were both fun and educational.

251. The training inspired me to explore deeper into the subject matter.

252. I was hoping for more opportunities for group collaboration.

253. The training materials were comprehensive and informative.

254. The trainer's industry insights provided valuable context to the content.

255. The training's hands-on labs were a highlight of the course.

256. The training helped me gain a deeper understanding of complex topics.

257. I found the training to be too theoretical for immediate application.

258. The trainer's anecdotes and real-world examples were engaging.

259. The training was well-paced and allowed time for participant questions.

260. The training content was highly relevant to current industry challenges.

261. I appreciated the trainer's availability for personalized guidance.

262. The training had a significant impact on improving my problem-solving abilities.

263. I was hoping for more opportunities for collaborative projects.

264. The training materials were well-organized and easy to reference.

265. The trainer's expertise and passion were evident throughout the course.

266. The training provided valuable tools and techniques for my work.

267. The training had a positive impact on my decision-making skills.

268. I was impressed by the trainer's dedication to participant success.

269. The training was a stepping stone towards achieving my career goals.

270. I found the training to be too focused on theory without enough practical exercises.

271. The trainer's enthusiasm for the subject matter was contagious and motivating.

272. The training was well-structured and easy to follow.

273. The training materials could have benefited from more real-world case studies.

274. I appreciated the trainer's interactive approach to teaching.

275. The training inspired me to explore deeper into the subject matter.

276. I expected more opportunities for group discussions and collaboration.

277. The training materials were comprehensive and informative, serving as valuable references.

278. The trainer's industry insights and real-world examples provided invaluable context.

279. The training's hands-on labs were a valuable learning experience.

280. The training helped me gain a more profound understanding of complex topics.

281. I found the training to be more theoretical than practical for immediate application.

282. The trainer's anecdotes and storytelling made the content engaging.

283. The training was well-paced, allowing for participant questions and discussions.

284. The training content was highly relevant to current industry challenges.

285. I appreciated the trainer's willingness to provide personalized guidance.

286. The training had a significant positive impact on improving my problem-solving abilities.

287. I was hoping for more opportunities for collaborative projects and teamwork.

288. The training materials were thoughtfully organized and easy to reference.

289. The trainer's expertise and passion for the subject matter were evident throughout.

290. The training provided valuable tools, techniques, and best practices for my work.

291. The training had a transformative impact on my decision-making skills.

292. I was impressed by the trainer's unwavering commitment to participant success.

293. The training served as a crucial milestone in my career advancement.

294. I found the training to be more theory-focused, lacking practical exercises.

295. The trainer's enthusiasm and passion for the subject were contagious and motivating.

296. The training was thoughtfully structured and easy to follow.

297. The training materials could have been enriched with additional real-world case studies.

298. I appreciated the trainer's interactive teaching style, which kept us engaged.

299. The training inspired me to delve deeper into the subject matter.

300. I expected more opportunities for group discussions, networking, and collaboration.

301. The training materials were thorough and served as valuable references for future work.

302. The trainer's industry insights and real-world examples were eye-opening and practical.

303. The training's hands-on labs were a pivotal aspect of the learning experience.

304. The training equipped me with a deeper understanding of complex topics.

305. I found the training to be more theoretical in nature, with limited practical application.

306. The trainer's storytelling and anecdotes were engaging and memorable.

307. The training was thoughtfully paced, allowing ample room for participant questions.

308. The training content was highly relevant to addressing current industry challenges.

309. I appreciated the trainer's accessibility for personalized guidance and support.

310. The training had a substantial positive impact on enhancing my problem-solving capabilities.

311. I wished for more opportunities to engage in collaborative projects and teamwork.

312. The training materials were meticulously organized and easy to refer back to.

313. The trainer's expertise and genuine passion for the subject matter were consistently evident.

314. The training provided a comprehensive toolkit of tools, techniques, and strategies for my work.

315. The training was a transformative experience that significantly enhanced my decision-making skills.

316. I was deeply impressed by the trainer's unwavering dedication to ensuring participant success.

317. The training played a pivotal role in shaping the trajectory of my professional journey.

318. I found the training to be more inclined toward theory, with limited practical exercises.

319. The trainer's enthusiasm and passion for the subject matter were highly infectious and motivating.

320. The training was meticulously structured and easy to navigate.

321. The training materials could have been further enriched with additional real-world case studies.

322. I valued the trainer's interactive teaching style, which kept us engaged and actively learning.

323. The training ignited my curiosity to explore the subject matter in greater depth.

324. I anticipated more opportunities for group discussions, networking, and collaborative endeavors.

325. The training materials were comprehensive and provided valuable references for future endeavors.

326. The trainer's industry insights and practical examples were eye-opening and highly applicable.

327. The training's hands-on labs were a pivotal aspect of the learning journey.

328. The training equipped me with a deeper understanding of intricate concepts.

329. I perceived the training to be predominantly theoretical, with limited practical application.

330. The trainer's storytelling and real-world anecdotes were captivating and left a lasting impression.

331. The training was thoughtfully paced, allowing ample room for participant questions and engagement.

332. The training content was acutely pertinent to addressing contemporary industry challenges.

333. I highly valued the trainer's accessibility for personalized guidance, support, and mentorship.

334. The training had a profound and lasting impact on enhancing my problem-solving abilities.

335. I had hoped for more opportunities to actively participate in collaborative projects and teamwork.

336. The training materials were impeccably organized and provided a wealth of references for future work.

337. The trainer's expertise and authentic passion for the subject matter were consistently evident.

338. The training provided an extensive toolkit of tools, techniques, and strategies that are readily applicable to my work.

339. The training was a pivotal catalyst for the significant improvement of my decision-making skills.

340. I was deeply moved by the trainer's unwavering dedication to ensuring the success and growth of each participant.

341. The training played an indispensable role in shaping and advancing my professional journey.

342. I perceived the training to be more theoretical in nature, with limited opportunities for practical exercises.

343. The trainer's infectious enthusiasm and fervor for the subject matter were inspiring and motivational.

344. The training was meticulously structured and easy to navigate, facilitating effective learning.

345. The training materials would have benefited from additional real-world case studies to enrich the learning experience.

346. I greatly appreciated the trainer's interactive teaching style, which fostered active engagement and participation.

347. The training ignited a desire to delve deeper into the subject matter, encouraging lifelong learning.

348. I had expected more opportunities for group discussions, networking, and collaborative endeavors.

349. The training materials were comprehensive, serving as valuable references for ongoing projects.

350. The trainer's industry insights and practical examples were illuminating and directly applicable.

351. The training's hands-on labs were an essential aspect of the educational journey.

352. The training provided me with a profound understanding of intricate concepts, enhancing my skills.

353. I perceived the training to be primarily theoretical, with limited practical application.

354. The trainer's storytelling and real-world anecdotes were captivating and memorable.

355. The training was thoughtfully paced, allowing ample room for participant questions and engagement.

356. The training content was highly pertinent to addressing contemporary industry challenges.

357. I highly valued the trainer's accessibility for personalized guidance, support, and mentorship.

358. The training had a transformative impact on enhancing my problem-solving abilities.

359. I had hoped for more opportunities to actively participate in collaborative projects and teamwork.

360. The training materials were impeccably organized and served as a wealth of references for future work.

361. The trainer's expertise and authentic passion for the subject matter were consistently evident.

362. The training provided a diverse toolkit of tools, techniques, and strategies that are readily applicable to my work.

363. The training served as a pivotal catalyst for the significant improvement of my decision-making skills.

364. I was deeply inspired by the trainer's unwavering commitment to ensuring the success and growth of each participant.

365. The training played a crucial role in shaping and advancing my professional journey.