UNIVERSITY
OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

Natural language search for personal information retrieval

Benefits and limitations to the user's experience in bookkeeping software.

Jeroen Smienk

January 31, 2023

Master's Thesis

Interaction Technology

University of Twente

Enschede, The Netherlands

Author

Jeroen Smienk (s2385597)

Master Interaction Technology

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)

University of Twente

Examination Committee

dr. Shenghui Wang

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), Human Media Interaction (HMI)

University of Twente

dr. Rúben H. De Freitas Gouveia

Faculty of Engineering Technology (ET), Interaction Design (IxD)

University of Twente

dr. Mariët Theune

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), Human Media Interaction (HMI)

University of Twente

Supervisors

dr. Shenghui Wang

M.Sc. Remco de Man (Moneybird)

M.Sc. Marijn Kleijer (Moneybird)

Acknowledgements

This master's thesis is my last step at University of Twente towards a degree in Interaction Technology. First and foremost, I want to thank my colleagues at Moneybird for thinking about an assignment together and providing me with the challenge, but also the resources to complete it. This includes a computer, a place to work among its employees, and financial compensation to participants of my user study. Second, for their interest and support, I want to thank my external supervisors Remco de Man and Marijn Kleijer for their weekly meetings and guidance. Third, the clients of Moneybird that participated in my user study who were willing to travel to Enschede for it. Without them, I could not have evaluated the natural language search prototype. Fourth, I want to thank my supervisor, Dr. Shenghui Wang from the Human Media Interaction research group at University of Twente for her biweekly guidance and feedback and also Dr. Rúben De Freitas Gouveia for meeting and discussing my user study design, despite having been occupied during those times. Lastly, fellow students that had already completed their master's theses, my friends, my family, and Romy for their continuous interest and support. Not only during the thesis, but my entire studies.

Abstract

Natural language processing (NLP) is rapidly progressing and more specific areas such as natural language understanding (NLU) and natural language generation (NLG) are becoming incredibly advanced. However, the use of NLU in personal information retrieval is limited and disadvantages or lack of advantages to the user's experience may have hindered its adoption. Studies on how natural language search affects the user's experience are scarce and dated. NLP in search may transform the norm of forcing users to cognitively transform their information need into a structured path or keyword query towards the answer. The goal of this master's thesis was to find the benefits and limitations of natural language search to the user's experience of personal information retrieval in bookkeeping software. A limited prototype was developed of a state-of-the-art natural language search interaction for search in bookkeeping software, based on the interaction and technical designs for natural language search in related work. A scenario-based user study was designed to evaluate how the prototype system helps or limits users with their information needs. The qualitative and quantitative data collected from nineteen participants was analysed to identify four major themes, assess the prototype's performance, and produce concrete learnings. The main benefits of natural language search are described in terms of providing a more intuitive way for users to express their information needs, including the ability to describe complex information, unfamiliar or unknown items, and domain-specific language. Limitations discussed include conditioning to keywords and lack of trust in the system's capabilities that come with increased cognitive effort, and additional physical effort for inexperienced typists. Adoption by expert users is also limited due to perceived greater efficiency of their common methods. Clear introduction, but especially building trust through successful performance are crucial for overcoming limitations. The major themes, benefits, limitations, performance assessment, and learnings are discussed in detail and the thesis is concluded with advice for future work and continued development.

Contents

List of Figures

List of Tables

List of Acronyms

API Application Programming Interface

IE Information Extraction

IR Information Retrieval

LLM Large Language Model

NER Named Entity Recognition

NLG Natural Language Generation

NLP Natural Language Processing

NLU Natural Language Understanding

POS part-of-speech

RDB Relational Database

RE Regular Expression

RNN Recurrent Neural Network

SER Slot Error Rate

SQL Structured Query Language

UI User Interface

UX User Experience

QA Question Answering

1 Introduction

The amount of digital data that is stored on personal devices and in the cloud is increasing and predicted to reach 200 zettabytes (trillion gigabytes) by the year 2025 [9]. This includes all photos shared online, emails sent, documents uploaded, and metadata stored during these interactions. Online data is often accessible to users through means of conventional keyword search and (faceted) navigation through the website's structure [42, p. 74]. Traditional Information Retrieval (IR) focused on the technical user, who has in-depth knowledge of the system, to find what they need [33, p. 432]. Despite efforts being made to make search more interactive and helpful using automatic completion, suggestions, and related searches [36, 42, p. 124], accessing these increasing amounts of data is largely stuck on keyword-based search and structured navigation. When searching for files on a personal device or for articles and information on the web, users are forced to cognitively transform their information need into a structured path or query towards the answer that is understandable to the service [2, 19, 43, p. 50].

The research area of IR primarily focuses on the efficient storage and effective retrieval of all this data. However, less focus is directed at the search experience, helping the user find what they need without making them think too hard about how they need to search for it. Many search functionalities require the user to enter keywords or click through structured menus, while intuitively, the most simple way to find what you are looking for may be to directly ask for it [42, p. 106]. Natural language describes the language we use to communicate with each other in day-to-day situations. Natural language understanding is starting to find its way to the wider public. Google search supports natural language Question Answering (QA), virtual assistants such as Apple's Siri can handle QA and simple commands, and customer support chatbots try to provide answers to frequently asked questions. However, these examples mostly focus on factoid QA instead of personal IR. Personal IR concerns your own data, such as your documents, emails, and contacts on your device or in the cloud. Since 2015, macOS Spotlight features a very limited natural language search functionality that enables its users to find "emails from May" or "PDF files from last month" [23], but asking specifically about either sent or received email or PDF files larger than a certain size is not possible.

It is unclear exactly why natural language search for personal IR is not implemented on a larger scale, unlike public QA, which suggests the availability of sufficient Natural Language Understanding (NLU) technology. These technologies may be too expensive or laborious to implement for small and medium enterprises that do not have teams dedicated to Natural Language Processing (NLP), which slows down adoption. Additionally, current state-of-the-art advancements are primarily machine learning-based, requiring large datasets, even for fine-tuning existing models. Besides, users are familiar with traditional navigation and search methods that have been around for decades and may not expect or understand the capabilities of natural language search. Despite natural language being fit for casual users and perceived as familiar, convenient, and guiding in a search context [27], users need to know what is possible to ask [13, 27]. Furthermore, the intended benefits of natural language search for personal IR may not outweigh its limitations, compared to traditional search and navigation. The few empirical experiments that have been conducted that address this are mostly dated and focus on web search and QA instead of personal IR [53, 27, 26, 29, 13].

Conversational search may transform the search experience into a version that interactively helps users find information that better fits their information needs, find it faster and with less cognitive effort. This master's thesis aims to find the usability benefits and limitations of interactive natural language search for personal IR. This is explored in the context of the bookkeeping software of Moneybird. Moneybird is a cloud-based financial management tool designed to assist small businesses and freelancers in the organization and automation of their financial processes. It follows the principles of double-entry bookkeeping, in which journal entries are used to record business transactions and maintain balance in accounts. Instead of allowing users to directly input journal entries, Moneybird generates them based on financial statements such as ledger accounts, invoices, and payments entered by the user or obtained through a banking connection. In addition to financial statements, Moneybird also includes other data types, such as contacts, projects, estimates, and products, which aid in the organization of the user's financial administration but do not directly affect their bookkeeping. Moneybird is built in *Ruby on Rails*[1] using *Ruby*[2] and web technologies *HTML*, *CSS*, and *JavaScript*. It is cloud-based, hosted on *Amazon Web Services*, and using relational database technology *PostgreSQL*[3] with *Elasticsearch*[4].

Currently, keyword search and navigational functionalities aim to satisfy users' information needs. Interactive natural language search may enhance data accessibility in Moneybird by letting users type their information need to ask for specific invoices or other documents and their financial performance directly. For example, by typing search

[1] version 6.1: `https://guides.rubyonrails.org/v6.1/`
[2] version 2.7.5: `https://docs.ruby-lang.org/en/2.7.0/`
[3] version 13.2: `https://www.postgresql.org/docs/13/`
[4] version 7.10: `https://www.elastic.co/guide/en/elastic-stack-get-started/7.10/`

requests such as "invoices that are due this month", or "KPN invoices of last year between 50 and 100 euros", or "how much VAT am I due this quartile?" Misconceptions about how keyword search works can constrain and confuse the user [2, 19]. For example, in a system using boolean search where a document qualifies either if it contains all keywords or if it contains at least one of the keywords. Meaning and intention are also not sufficiently expressed using keywords alone. The query "invoices march due" may find documents that include the text "invoices", "march" and "due" (or any of them), instead of being due in March, missing crucial semantics. The user may even intend to find invoices that are sent in March and are already due. Search that utilizes NLU may infer these alternatives and offer them, while expressing the query in natural language resolves these ambiguities completely.

Benefits and limitations of typing a search query in natural language for personal IR are unknown or based on outdated studies. Technology advances quickly and its state has a significant impact on the experience of the user [27]. Understanding the users' experiences with a natural language search interaction for personal IR can help understand the barriers to adoption of the technology. Therefore, the goal of this master's thesis is to find benefits and limitations to the user's experience of natural language search in Moneybird's bookkeeping software compared to its traditional search functionality. The main research question that this work aims to answer is:

"How does a natural language search interaction help or limit the user's experience while resolving their information needs in bookkeeping software compared to traditional search?"

The following list of subquestions focuses on answering smaller problems needed to answer the main research question:

1. *"What is a state-of-the-art interaction design of a natural language information retrieval system?"*
   (a) *"What search features and natural language features belong in a state-of-the-art natural language search interaction?"*
   (b) *"How can the natural language search system be integrated into Moneybird without impeding the traditional search interaction?"*
   (c) *"How can the system's capabilities and expectations be conveyed to the user?"*
2. *"How can a prototype of the interaction design using Dutch natural language be realized in the context of Moneybird?"*
   (a) *"Which existing technologies can be used to implement the design?"*
   (b) *"How does the system fit into Moneybird's architecture?"*
3. *"How can the prototype interaction be evaluated and compared to traditional search?"*

The main research question is answered by the analysis of an on-location user study conducted with clients of Moneybird. This involves designing, developing, and piloting a prototype of a natural language search interaction accepting text input. Subquestion 1 is answered by designing the interaction based on modern search practices and previous work on conversational search. The interaction design considers the features important to a clear and useable natural language search interaction. Subquestion 2 is answered by designing the technical implementation of a prototype of the interaction design based on related work and the author's experience with software engineering. The technical design considers the available technologies and desired architecture to implement and integrate the prototype within Moneybird. Subquestion 3 is answered by designing the user study based on related work.

Chapter 2 discusses the background required to understand this work. Chapter 3 then outlines the related work that lies at the basis of the following chapters: the interaction design of a state-of-the-art natural language search interaction in Chapter 4 and the technical design of the prototype in Chapter 5. Chapter 6 details the user study and prototype evaluation designs, and discusses learnings from piloting the experiment. Chapter 7 reveals the results of the user study, analysed according to the study's design, discusses the prototype evaluation, and lists improvements to the interaction design and implementation. Chapter 8 concludes the research, acknowledges limitations, and mentions advice for future work.

2 Background

Personal Information Retrieval involves the user's own data, such as documents and emails, instead of new information found in a web or library search. We can partially recall details of our own items because we have encountered them before [43, p. 62]. These details include temporal information, such as time of latest use, or when something was sent or received, and are not typically supported by standard interfaces. Because personal items are managed by the user, the efficacy of the search depends on their own data storage strategy. However, it is doubtful how constructively people manage their information space and how willing we are to devote time to create useful structure by using folders and hierarchies (e.g., [55]).

2.1 Relational Databases

Moneybird uses a Relational Database (RDB) to store the user's financial data. An RDB organizes data into tables, where each table represents a distinct class of object or a relationship between objects. Each row in a table represents a single item, and each column represents a characteristic of that type of item. Two important concepts in an RDB are 'primary keys' and 'foreign keys'. A primary key uniquely identifies the object in the table, whereas a foreign key refers to the primary key of another object in another table. In the context of Moneybird, all journal entries of each users' administrations are stored in a single table that can be distinguished by the administration identifier foreign key. Other data types, such as contacts, ledger accounts, documents, tax rates, products, and goals, are also stored in tables with an administration identifier foreign key. In the *Ruby on Rails* framework, models of these data types are mapped to the tables in the *PostgreSQL* database using the Active Record architectural pattern [17, p. 160]. The Active Record architectural pattern is a design pattern used in software development to represent the database tables as classes in an object-oriented language. In this case, *Ruby*. The Active Record pattern allows objects to be created, retrieved, updated, and deleted from the database without writing raw Structured Query Language (SQL) queries, but by chaining method calls. This uses an object-relational mapping (ORM) to simplify database interactions by abstracting away complexities of working with a database.

2.2 Search and User Experience

Search is an iterative process of keyword formulation, reviewing the results (or a preview thereof), and reformulating the query to resolve an information need [2, 19, 43, p. 53]. Creating an accessible and intuitive search functionality also relies on its usability, or User Experience (UX). The system needs to be clear about what it expects from the user and what results are returned and why. Traditional information retrieval often involved professionals with understanding of how queries need to be phrased, but search functionalities and expectations need to be clear to the non-technical user now [33, p. 432]. Children who are not conditioned to use keywords experience difficulties with applying the rules of this search paradigm [26]. Children who have not learned to search using keywords have trouble choosing initial terms, refining them, and understanding the relation between the chosen terms and the results, while some rather type an entire question into the search box.

Modern search engines offer a range of features to help users with query formulation and make searching easier. Search engine result pages explain hits by highlighting relevant parts of the results that match with the search query. Automatic completion suggestions save time and help recognition by suggesting possible completions of the remaining query [36, 42]. Recognizing previous experiences is easier than recalling them from memory. This characteristic is called *recognition over recall* [42, p. 109]. Completion suggestions can be based on popular searches collected from the other users or previous searches by the same user, powered by encoder-decoder architectures used to offer fluent less restrictive completion predictions [34]. Instant search provides initial results while the user is typing [42, p. 112], which saves them time and allows them to adjust or further specify their query if the initial list is not reflecting their needs. Related searches help the user get inspiration or clarify their query [42, p. 120]. Related searches could be based on related elements in a knowledge structure, or on popular queries among other users.

2.3 Natural Language Processing

Natural Language Processing (NLP) and its areas of research concern themselves with processing "natural language", mixing computer science and linguistics. Natural language refers to the ordinary language people communicate with every day, all over the world. NLP tries to process the language to analyse bodies of text or short sentences for computer automation. Common applications include grammar and spell checking, typing prediction, spam filtering, summarizing news articles or research papers, and machine translation. Practically, everything involving 'human text'. Since the 1950s, research on this topic focussed on different areas of applications from machine translation, interfacing with databases, and question answering [3], to Weizenbaum's famous *ELIZA*

program, mimicking a Rogerian psychotherapist [54]. Within the domain of NLP lies a field of research that tries to attach meaning to the words and sentences that make up our languages. Research in the area of Natural Language Understanding (NLU) tries to comprehend what the writer intends to say to better interpret their judgement, emotion, or sentiment; categorize or order bodies of text; and answer questions or follow commands. Typical applications include film and product review analysis, analysis of online public opinion, and chatbots and smart assistants such as Apple's *Siri*, and Amazon's *Alexa*. Natural Language Generation (NLG) in turn, concerns itself with generating human-readable text that can be used to communicate back to a user in natural language. Systems employing these technologies can have intelligently appearing interactions with their users. As Bates put it: "Language is so fundamental to humans, and so ubiquitous, that fluent use of it is often considered almost synonymous with intelligence" [3, p. 1]. However, the endless complexity and ever-changing nature of language makes it difficult for computers to deal with. NLP is mostly used in search to find better matches with less effort and restrictions for the user. Ranking performance is improved by addressing challenges such as misspellings, rare words, synonyms, and disambiguation.

State-of-the-art approaches to many different tasks in NLP utilize neural networks, including spelling correction [20], generating automatic completion suggestions [20, 34], Named Entity Recognition (NER) [57], calculating word embeddings [56], entity linking [50, 35], semantic parsing [12, 24, 59], question answering [32], intent determination and slot filling [40, 21, 28], and conversational request reformulation [20]. An *embedding* is a vector representation of the meaning of a word computed from their distribution in texts (i.e., its context) [25, p. 102]. Similar words are represented by similar vectors in multidimensional space. Static embeddings exist and contextual embeddings represent words in context and are recently calculated using bidirectional encoders [56]. Neural networks based on convolutions or recurrence are designed and trained or existing multipurpose language models are fine-tuned. Fine-tuning means training a general purpose model for a specific task. This has the benefit of not needing as many labelled samples. Large Language Models (LLMs), such as *BERT* [11] and *GPT-3* [6], are pre-trained neural network models successful at a wide range of NLP applications. They utilize an encoder-decoder structure, but do not use convolutions or recurrence. Instead, these models are based on the novel transformer architecture that can handle long-range dependencies better. These foundational models can be fine-tuned to a specific application and deliver state-of-the-art performance on a variety of tasks that would have traditionally necessitated the development of separate models. However, rule-based, statistical, or hybrid approaches remain popular in specific use cases because these offer high precision and can cover enough subjects in narrow domains [25, p. 536]. They also do not require as large a dataset of samples to train on. These approaches use a variety of tools, such as regular expressions, Conditional Random Fields (CRFs) (e.g., [10, 35]), or *semantic grammars* (i.e., a Context-Free Grammar (CFG) that uses semantic entities or slot names on its left-

hand side instead of POS) [25, p. 536]. Selecting one or a combination of these methods is relevant to the technical design of a natural language search interaction.

Some information in natural language text is better understood or processed if it is first extracted in its appropriate format. For instance, mentions of events, people, and locations, relations between entities, numeric information such as money and quantifiers, and temporal expressions such as "yesterday", "this quartile" or "May 30th". The process of extracting these kinds of limited semantic content from text is called Information Extraction (IE) [25, p. 363]. IE can be done using several categories of algorithms, including handwritten patterns or rules, supervised machine learning, semi-supervised machine learning, and unsupervised machine learning [25, p. 367]. Specific benefits and disadvantages determine the best approach for a system. Handcrafted patterns or rules offer high precision and can be precisely tailored to specific domains, but often suffer from lower recall and are expensive to create. On the other hand, supervised machine learning assumes enormous datasets with plenty of annotated samples that are also expensive to create if non-existent. Unsupervised methods do not use annotated training data and try to find relations from texts on the web or corpora. For instance, by using verbs and sentence composition to connect a name and birthdate in a biography.

Named Entity Recognition (NER) finds sequences of text that are proper names, including people (e.g., "Jane Austen"), locations (e.g., "Enschede"), and organizations (e.g., "University of Twente") using a trained sequence labeller [25, p. 164]. CRFs were the state-of-the art technique before deep learning techniques gained traction [35]. A CRF is a probabilistic classifier for segmenting and labelling sequential data, making it a useful technique for NER [10]. However, state-of-the-art NER is also achieved using fine-tuned LLMs [51] using frameworks such as *Flair* [1]. Times and dates can be represented in many written forms and need to be *normalized* to a common format to reason or compute with [25, p. 375]. Temporal expressions can refer to an absolute point in time (e.g., "30 May 1997"), but also a relative point in time (e.g., "last week"), and durations with a mix of these (e.g., "between 2010 and now"). Relative points in time are implicitly anchored. For example, to another point in time mentioned earlier in the text, to the present in speech, or to the date of the document in the case of a news article. A common challenge is not tagging false positives such as "2001: A Space Odyssey" [25, p. 376]. Modern approaches to tagging and normalizing temporal expressions are still rule-based and can be extended to work for many languages [25, p. 377] (e.g., [8, 47]).

2.4 Frame-based task-oriented dialogue systems

A natural language search interaction can be compared to how dialogue systems such as virtual assistants help their users with information needs. These systems are also known as task-oriented agents [25, p. 521]. Unlike chatbots that try to mimic human to human

interaction, task-oriented agents do not intend to have extended conversation with the user. Classic task-oriented dialogue systems use a knowledge structure called 'frames' to represent possible user intentions and store the information from the conversation that is needed and already extracted into a frame's 'slots' [25, p. 535]. For example, a flight booking frame might include slots for the origin city, destination city, departure date and time, and arrival date and time. A slot may also be a frame itself, which forms a hierarchical structure of required information to be filled. The request can only be fulfilled if all slots are supplied. Before a frame can be discussed by the agent, the dialogue's domain and the user's intent must be determined to select a frame. Systems working in a single domain only have to determine the user's intent. Intent determination and slot filling is achieved with NLU.

Traditional systems employed handwritten rule-based approaches for intent determination and slot filling using regular expressions or semantic grammars [25, p. 536] (e.g., [30, 38]). Before LLMs, state-of-the-art systems utilized bidirectional Recurrent Neural Networks (RNNs) with gate and attention mechanisms to view the problem as a sequence labelling task (e.g., [40, 31, 60, 21]). These models outperform models used in previous work, including CRFs, Support Vector Machines (SVMs), and regular RNNs. Joint models outperform separated models, indicating that one task promotes the other if in one model. Sentences are annotated with the correct set of slots using slot tags, and domain and intent using special domain and intent tokens at the end of the sentence. The intent is classified using the neural network and slots are filled using the tagged tokens. With sufficient training samples, these models outperform original rule-based approaches, but may not offer higher precision in complex cases such as negation.

Modern systems, such as Apple's Siri and Amazon's Alexa, use control structures designed around an advanced version of the frame, a dialogue-state architecture consisting of several components [25, p. 538]: Automatic Speech Recognition (ASR) to transform the user's utterance into a textual representation, NLU to determine the intent of the user and extract the necessary context, a Dialogue State Tracker (DST) to keep track of the conversation, Dialogue Policy to decide what to reply, NLG to construct the textual representation of the response, and Text-to-Speech (TTS) to generate a spoken utterance.

Chatbots and task-oriented agents often employ *implicit confirmation* by restating some information provided by the user in their response to confirm the system understood the correct intention [25, p. 543]. For example, by replying "I found three restaurants that are cheap, Indian, and within 5 kilometres" to a search for restaurants. Similarly, *progressive prompting* or *escalating detail* can help provide instructions to the user on how to use the system [25, p. 544]. For instance, instead of replying "I don't understand" when an input is not understood completely, the system could elaborate on what parts it did not understand or reiterate what users may ask for.

3 Related Work

In their theoretical framework of conversational search, Radlinski and Craswell define conversational IR as "a system for retrieving information that permits a mixed-initiative back and forth between a user and agent..." [39, p. 120] They describe a model for chat type interfaces that allows the user to make a natural language request, proposes suggestions or results, may ask the user for clarification, and takes feedback. The model expects and is capable of certain actions, and has several properties:

- User Revealment: the system helps the user discover and express their true information need. For example by showing a set of items and their properties which may help users recognize additional differential attributes to use in request reformulation.
- System Revealment: the system informs users of its capabilities to set or adapt their expectations. For example by confirming understanding or requesting a critique (a critique gives feedback on important aspects and explains a modification to the results, such as "but then from this year").
- Mixed Initiative: the system and user may both take initiative.
- Memory: the user may refer to previous statements.
- Set Retrieval: the system may reason about sets of complementary items.

From the perspective of IR, an important role of conversation is determining a mutual understanding of what the inquirer asks and what the answerer can provide [39]. The authors mention personal information search as an example application where users have a range of contextual information in mind and an effective conversational search system should aid formulation of this information need without having the user remember a complex query language. A mixed-initiative system may narrow down the search space by offering choices when appropriate while allowing the user to express themselves freely in text, and system-initiated refinement suggestions may lead the user to remember other attributes they wish to provide. The interaction designed in this thesis adheres to their model in many ways: helping users reformulate their request, revealing its capabilities, taking initiative to suggest refinements depending on the current set of results, accepting

critique as a follow-up request referring to the previous request, and suggesting completions to narrow down the search space and inform about its capabilities. The major difference is that this design is not meant as a chat or a system to have extended conversation with. Besides, the ability to provide feedback to the system is not incorporated in this design.

In 2005, Wang *et al.* [53] observed users tasked with finding information about *eBay*'s services via the keyword-based search functionality on their website and their natural language chatbot. Their main finding shows that task complexity was a major determinant in what system was perceived as better. On complex tasks, keyword-based search was perceived as easier and more enjoyable, and participants felt more confident. On simple tasks, this all applied to natural language search. Actual search performance of participants did not differ, but participants believed they were more successful with keywords. Some mentioned how the chatbot seemed smart when it replied with questions for confirmation of its interpretation of the query. Its major limitation is how dated both the study and the technology used to provide the chatbot with its capabilities are.

In 2010, Kaufmann and Bernstein [27] conducted a usability study among 48 users testing four interfaces for factoid QA. The interfaces ranged from natural language to more formal query input. The interface that was best-liked and perceived as most useful was one that allowed full English questions with a limited set of sentence beginnings. It was significantly preferred over keywords, a menu-guided, and graph-based query language. The authors mention how this contradicts previous work where keyword-based search was preferred among students. However, that previous work quoted inferior result quality of the natural language system as a reason. The authors conclude that the structure of natural language was perceived as familiar, convenient, and guiding, and is fitting for casual or occasional users.

In 2014, Li and Jagadish [29] developed a natural language query interface for relational databases. It explains to the user how their query is processed and generates multiple interpretations when ambiguities exist to interactively resolve them. The paper mentions how even naive users completed logically complex query tasks.

In 2018, Dubiel *et al.* [13] found that free form conversational systems that remember previous context or answers are experienced as more user-friendly and less cognitively demanding for complex tasks than single utterance slot-filling formats. However, single utterance systems are perceived as more predictable and cause less concern regarding the system's ability to accurately understand the input.

In 2018, Zhang *et al.* [61] describe the 'System Ask User Respond' paradigm in an e-commerce setting based on the belief that the system should ask questions actively to find out the user's information need. The user initiates a request and based on initial results, the system considers asking a question to narrow down the search results. The

conversation is led by the system and the user simply responds until the system is confident enough to show a short list of candidate products. The interaction designed in this thesis uses a similar loop, but is led by the user instead. The system does not steer the direction but offers contextual refinement suggestions. Their work is mainly a technical paper and also focuses on search in e-commerce. This area concerns unknown items, which is a more exploratory process than known-item retrieval.

In 2020, Rosset *et al.* [41] also mention that many users still use keywords because of years of experience with search systems that did not support natural language questions. Therefore, users must be subtly encouraged to ask natural language questions, now that these can be handled more effectively. Additionally, the system could lead the conversation by suggesting related natural language questions that the user may wonder about next. The work uses fine-tuned LLMs *BERT* and *GPT-2* to rank and generate suggestions, respectively. However, it is mainly a technical paper to generate useful and engaging suggestions, evaluating the effectiveness of the models, but it does not evaluate the effect on a user's natural language use or experience.

Besides the above work, the designed interaction and flow is mainly influenced by Guo *et al.* [20]. They showcase research on conversational search on the LinkedIn platform regarding help centre QA. Their high-level workflow is an iterative process that incorporates out-of-scope, incomplete, and ambiguous question handling, providing clarifications, and asking follow-up questions, until the user is satisfied with the result. The interaction design detailed in the next chapter combines this flow with modern search features discussed in the previous chapter, such as instant search, automatic completions, and related search suggestions. Guo *et al.* [20] used sequence-to-sequence neural networks to implement these features, including spelling correction, automatic completions, related search suggestions, automatic query expansion, and conversational query reformulation (handling critique).

The technical design of the prototype is mainly inspired by task-based dialogue agents and legacy NLU approaches by Lim and Lee [30] and Plachouras *et al.* [38]. The reason for choosing a more traditional approach is detailed in Chapter 5. Plachouras *et al.* [38] interact with an online database of macroeconomic data using natural language to search public online financial data. It is capable of answering well-formed English questions such as "What is the GDP of India in 2010?". The natural language input is 'understood' by first tagging entities that are likely to be important to the interpretation using NER. The authors only recognize a curated list of domain-specific entities. Intents are formed from sets of found entities and are scored using the entities' given scores and prevalences based on web counts. Intents are mapped to query plans by comparing the entities of the intent to the plan's requirements. A query plan is a set of search and filter instructions to find an answer. The results are communicated as a natural language answer to the question.

The method is purposely not grammar-based to be more flexible and domain-adaptable. The authors do not mention how entities are defined.

Lim and Lee [30] designed a rule- and template-based task-oriented agent that chains together predefined web services to answer multi-sentence natural language requests. To create a candidate service list, available web services are compiled by semantically annotating them and describing the related natural language expressions. It determines a workflow of actions by analysing each sentence based on its verbs and nouns, then mapping these words to concepts in one of three ontologies. An ontology is a way of organizing and representing concepts, properties, and relationships within a particular subject area. An example ontology about geometric shapes is shown in Figure 3.1. Using fuzzy string matching [37] or word embeddings (which can resolve synonyms and hypernyms [35, 32]) are two ways of matching user input to concepts in an ontology. Lim and Lee [30] propose three distinct ontologies to capture relevant concepts: Action, Object, and Input/Output. These contain concepts such as 'reserve', 'flight', and 'departure location', respectively. The concepts in the Input/Output ontology are similar to typical slots in a frame-based architecture. Verbs and nouns in the request are mapped to concepts in either ontology using natural language representations listed with each concept. Each sentence block is analysed to select the most similar service from the candidate list that best reflects the user's intention. The similarity is scored using an equation containing mapped ontology concepts, the number of concepts used by a service, and the number of concepts detected in the sentence. Scoring predefined services can be directly compared to scoring the similarity of intents modelled as frames. No details regarding the implementation of the service modelling and ontology mapping are discussed by the authors and user experience considerations are not mentioned. However, potentially resolving alternative interpretations by an additional user interaction is noted.

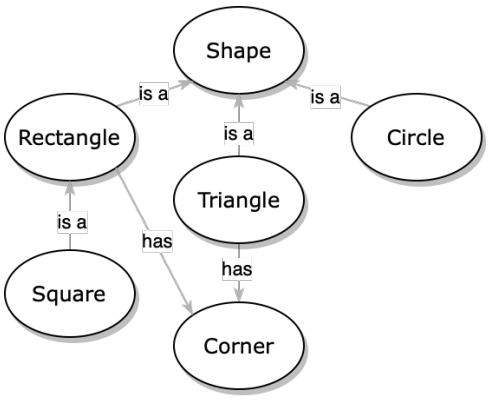

Figure 3.1: Limited example ontology on the subject of geometric shapes.

4 Interaction Design

Finding information on your personal device or personal environment in the cloud using natural language text input requires an interaction design that clearly conveys the system's capabilities and its expectations [27, 39, 13]. Eliciting natural language use is a challenge for a system expecting typed input, whereas this comes much more naturally with a spoken dialogue system [49]. The interaction must convey that users can ask for something the way they would ask another person. This chapter details how the design of a state-of-the-art natural language search interaction could be, based on modern search practices and related work on conversational search. The interaction design consists of the system's functionalities, its graphical User Interface (UI), and how users are expected to interact with it. The designed interaction is in Dutch because Moneybird's main interface language is Dutch.

Figure 4.1 shows an image of the UI of Moneybird's current search functionality before and during typing a keyword query. The search box floats on top of the current page when opened and features a wide search bar with placeholder text to suggest searchable areas within the application. The middle portion shows instant search results grouped per category while the user is typing. Text that corresponds to the query is highlighted and the first result is automatically selected for use with keyboard navigation. The footer shows a legend of keyboard shortcuts to navigate the search box with.

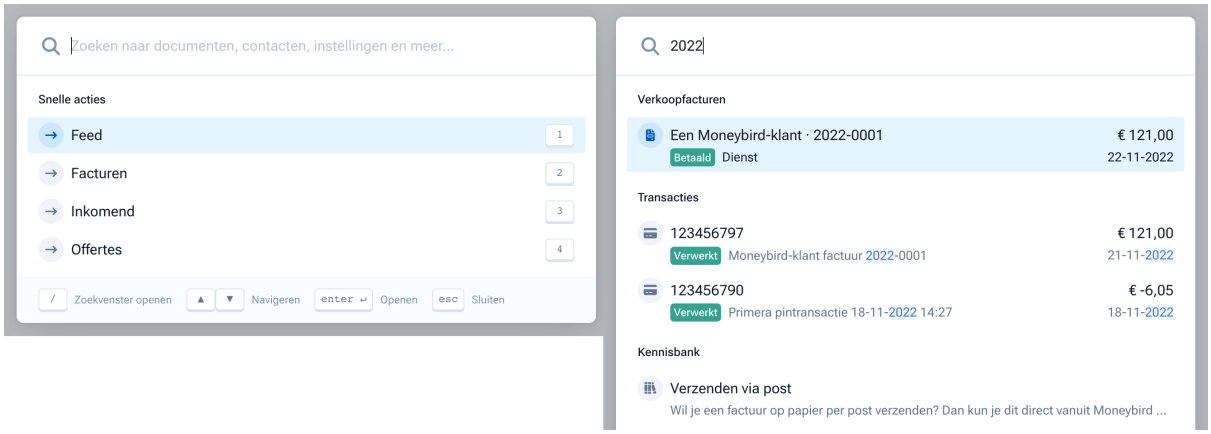

Figure 4.1: Dutch UI (cropped) of Moneybird's search functionality before entering a query (left) and showing instant search results as a categorized listing (right).

The objective of the proposed natural language search interaction is to fulfil a user's bookkeeping information needs that are formulated as typed natural language requests. Either by providing the correct results from the user's administration or, if not (directly) possible, iteratively helping them to the result using interactive feedback, completion suggestions, and related search suggestions. The interaction is a form of conversational personal IR where the user iteratively enters a natural language phrase and the system supports request creation during and after formulation by providing natural language responses [39]. This process is based on related work but most influenced by Guo *et al.* [20] and visualized in Figure 4.2.

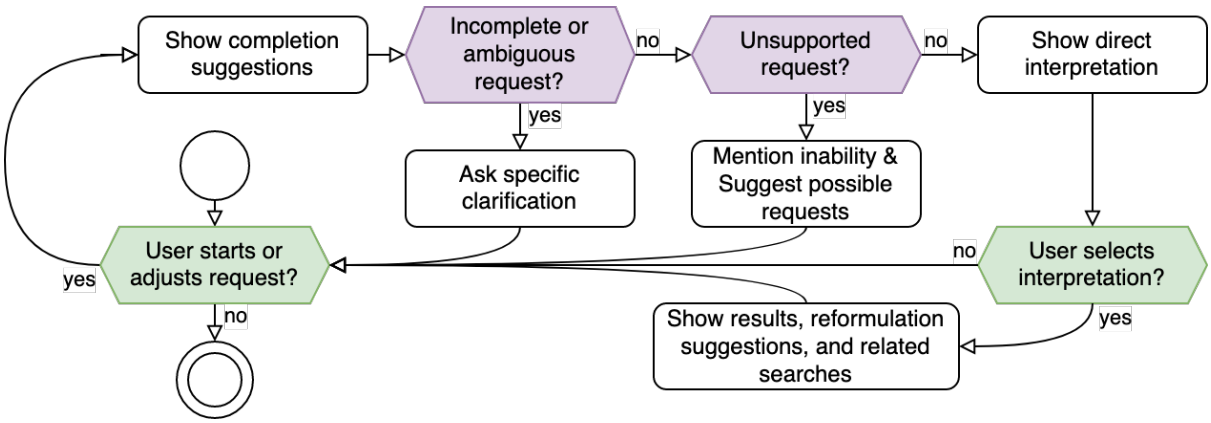

Figure 4.2: Diagram of the interaction flow of the proposed natural language search interaction.

The interaction is supported by several NLP features from modern search systems and research on conversational search. Before the user starts typing, a placeholder list item shows a random natural language request suggestion (e.g., "*paid invoices greater than 300 euros*") to communicate capabilities and expectations [42, p. 101]. During typing, the interaction offers completion suggestions to support request formulation [43, p. 53, 42, p. 124, 20]. Completion suggestions during request creation are possible because the system deals with requests that are typed instead of spoken. Typed input also avoids many challenges that are present in spoken dialogue systems, such as pauses, interruptions, mishearing, and understanding homophones [49]. Instant search results combine keyword search and NLU by showing the interactive feedback as a separate category above the categorized keyword hits. Instant search can facilitate a more interactive dialogue and aid quicker recognition of known-items in personal information retrieval [42, p. 114]. The interactive feedback shows up as a search result list item and uses natural language to reflect if and how it understood the user's request to build trust [18] using *implicit confirmation* [25, p. 543]. For example, the system may respond to a user entering "*invoices I still have to pay*" with the direct interpretation "*Unpaid purchase invoices.*" This interpretation can be selected to show the results by clicking on it. If the user selects a completion suggestion or the direct interpretation, the results will be retrieved and displayed. The interpretation also acts as a multi-document summary to clarify the rele-

vance of the potential results at a glance [43, p. 56]. Incomplete or ambiguous questions would be interactively clarified by asking for missing context [39, 61], while out-of-scope questions are identified to inform the user about the system's capabilities [39, 20]. This strategy is also known as *progressive prompting* [25, p. 544]. An additional benefit of combining NLU and keyword results is that the system does not have to determine if a string of text is meant as a keyword search query or a conversational search request.

Results are sorted by latest date first because this helps re-retrieval of personal items [14]. Results always come with a natural language result summary similar to the direct interpretation described above. Providing natural language answers and clarifications leads to the impression that the system understands the user and is perceived as signalling confidence about the retrieved results [27]. Attributes of results that are relevant to the request are highlighted to explain why these items are retrieved [42, p. 132]. Attributes of the retrieved documents that were not used in retrieval are suggested for request reformulation to narrow down results and make users aware of the support of these attributes [39, 20]. The interaction responds to conversational follow-up requests used to clarify or adjust a previous request (critiquing) [39, 20]. This prevents the user from typing the majority of the request again. This also means a request does not have to be formed in a single utterance to lower cognitive demand of request formulation [13]. Related searches suggest alternative requests that are similar to the one entered by the user to inspire request reformulation and introduce supported capabilities [42, p. 125, 39, 20]. These interactive assistances are specific to the situation and reactively provided, instead of being general search advice that has to be requested [43, p. 76].

Wireframes in Figure 4.3 below show the general layout of the designed UI. The left wireframe shows the interaction while the user is typing a request with completion suggestions below the search input and, in the body below that, the combined integration of the direct interpretation and the keyword results. The wireframes do not show actual keyword results. While the user is typing, instant keyword search results are loaded and the direct interpretation of the request is displayed on top. The first result list item is automatically focused (highlighted) to use with keyboard navigation as in the current situation (moving between list items with the up and down arrow keys and selecting one with Enter). The right wireframe shows the interactive feedback responding to an unsupported request. When the system does recognize that a request is not yet supported, it may reply: "*I do not understand. I may not yet be able to do what you ask. You can ask for invoices and key figures. For example, revenue in a certain period.*" The interactive feedback is again displayed as the first result list item, but it is not actionable and therefore the next result is automatically focused.

Selecting the interpretation presents the results of the request in the same floating search box. Different visualization methods are used depending on the type of result to provide the most appropriate format per information type. Personal data in the user's financial

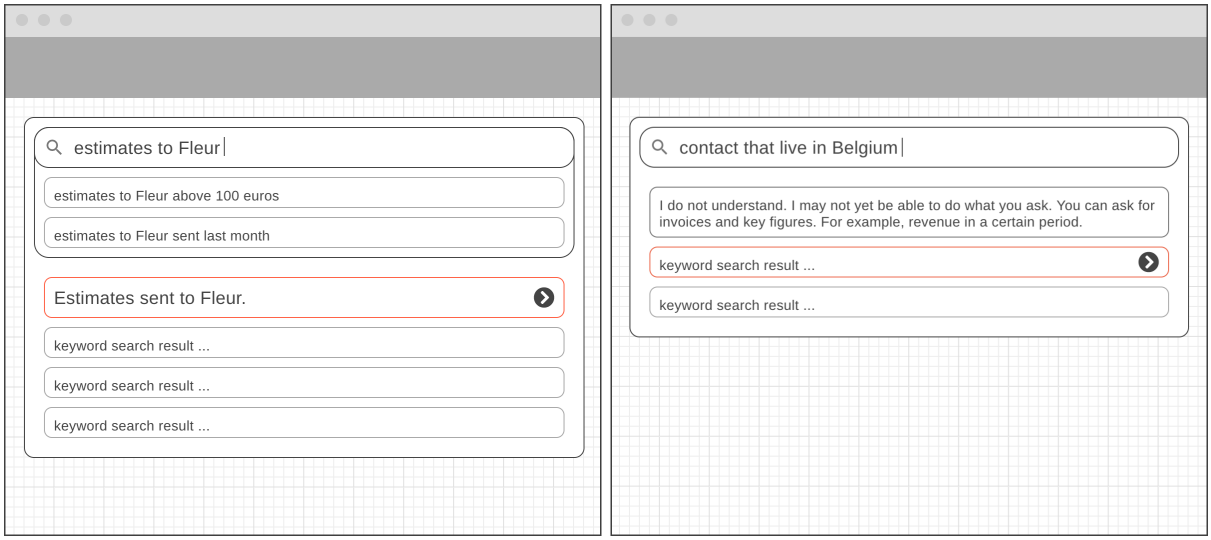

Figure 4.3: Two UI wireframes of the search box with integrated natural language request understanding.

administration that can be represented as a single data point (e.g., revenue of this month) is communicated with a textual response. Data represented as multiple points or time series (e.g., profit over the past year) is shown in a table and plot. Structured documents (e.g., invoices, estimates, and projects) are shown in a list using their attributes, such as title and date. If an error occurs, a request is not supported, or no results match the request, it is also explicitly communicated instead of leaving the user to guess [42, p. 148]. These three data modalities are summarized in Table 4.1.

Table 4.1: Overview of result modalities per result data type.

Modality	**Use case**
List	Structured documents
Text	Single data point
Rich (table and plot)	Multiple data points

The wireframes in Figure 4.4 show the general layout of the UI for the List and Rich result visualization methods. All results always include a textual response, whatever the main modality is. This is used to introduce or clarify results and help users immediately understand how the system interpreted the request. It is also beneficial on smaller devices where tables or plots may not be convenient, or when the intention is to communicate the answer via a speech-enabled device [38]. The left wireframe in Figure 4.4 shows the result page of a request using the List modality. The natural language response summarizes how the request was understood and what its results are. The wireframe does not show actual invoices, but when showing actual results, the due date of the invoices are highlighted using boldface to indicate their relevance to the request. A suggestion below the search

bar advises the user to include certain information to narrow down the results. The right wireframe in Figure 4.4 shows the layout of the results page for a search request regarding last year's revenue. The actual results would show a plot and table of past revenue performance where the wireframe shows placeholder illustrations. The search input now reads "*what about 2020?*" and illustrates how the user may subsequently input critique with the intention to adjust the current results.

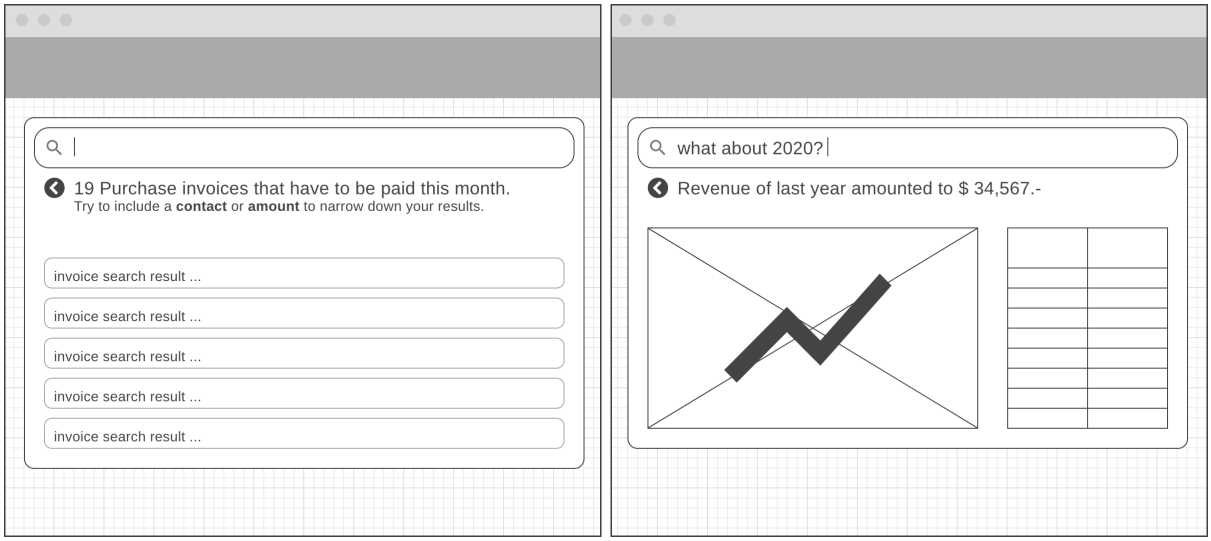

Figure 4.4: Two UI wireframes of the conversational search results page, displaying the List modality (left) and Rich modality (right).

5 Technical Design

The technical design describes the architecture and technologies used to implement a limited prototype of the designed interaction that can be used to evaluate the user's experience. It takes into account the scope of the prototype, the available resources, and the fit with the Moneybird application. The chapter outlines the system's architecture, its inner components, and specific algorithms designed to implement the prototype.

A prototype of the proposed interaction is realized using a technical design inspired by the natural language search interactions of Lim and Lee [30] and Plachouras *et al.* [38] and the architecture of frame-based task-oriented dialogue systems [25, p. 535]. Types of searches are modelled as predefined intentions, where each intention can be handled differently according to a corresponding frame. An intelligent component determines the intention based on the user's input and the search component handles the response determination, query construction and execution, and answer generation. The architecture is visualized in the diagram in Figure 5.1.

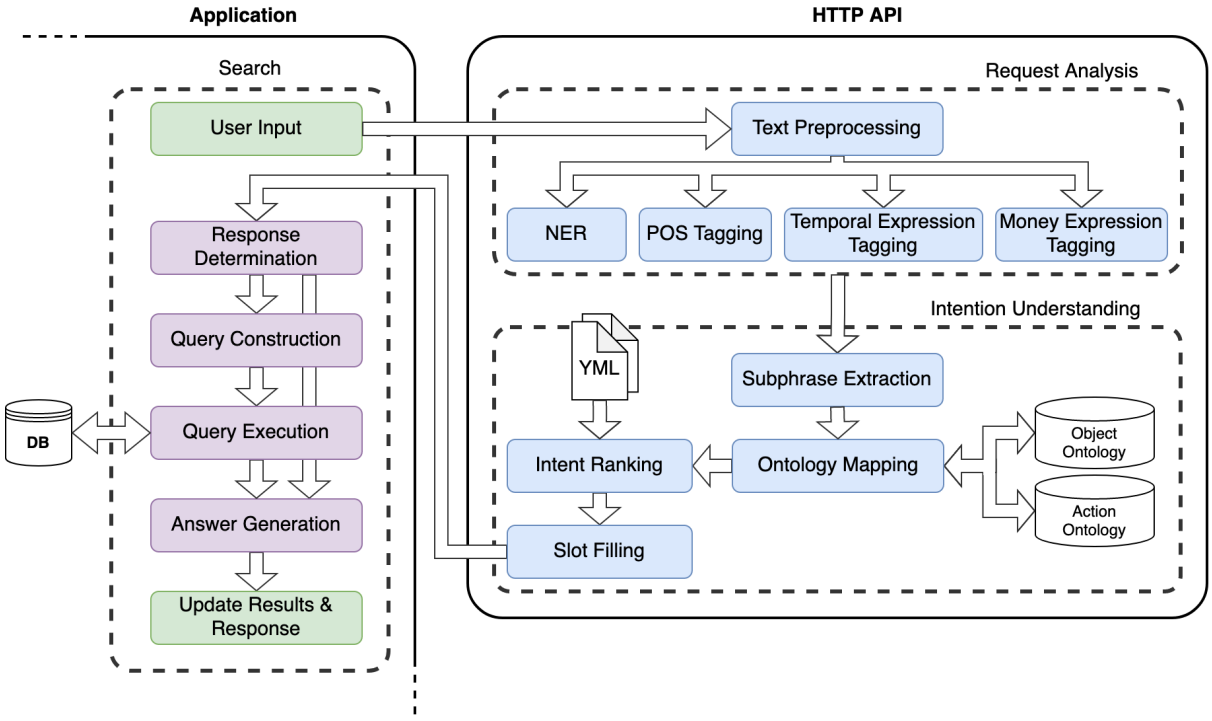

Figure 5.1: Diagram of the designed architecture of the prototype system.

Summarized, typed natural language search requests are analysed to score and rank predefined *intents* and conditions as frames and slots. The search request is divided into several parts called subphrases using its dependency tree. These subphrases are mapped to concepts in two domain ontologies containing natural language representations. The mapped concepts are matched to predefined intents and each available intent is scored using several designed functions. The search component selects the highest ranking intent to fill in the corresponding frame and its slots using the matched ontology concepts. A query is constructed and executed to retrieve the relevant data from the RDB. The frame includes the logic to compose a natural language response for the user. Request analysis, intent determination, and slot filling is done using a Python HTTP Application Programming Interface (API) called by the search component of Moneybird. The search component handles response determination, query construction and execution, answer generation, and presentation and is implemented in Ruby (on Rails).

5.1 Scope

Despite the availability of more modern and advanced technologies to implement intent determination and slot filling (e.g., machine learning-based intent classifiers and fine-tuned LLMs), this ontology mapping and intent ranking approach is selected because it does not require a set of labelled example search requests. Training or fine-tuning a neural network requires a dataset that does not exist for this context. The request understanding component is purposefully separated from the search component to allow upgrading its intent determination and slot filling performance without a major overhaul to the search component. For example, when new technology is available or a dataset is created.

The interaction design introduced a state-of-the-art natural language IR experience. However, the available time and resources do not allow for the complete implementation of this design, and that also affects the architecture. The described architecture was influenced by the fact that no applicable dataset of Dutch bookkeeping search requests was found. This ruled out neural network or fine-tuning approaches used for such tasks in current state-of-the-art implementations and steered towards the described approach. Some features, such as automatic completions and automatic corrections, are more straightforward to implement using a machine learning approach or generative LLM. Therefore, in combination with the available time, a subset of features and intents is chosen to develop a prototype for the user study. The list of features is shown in Table 5.1. The prototype integrates most features described in the interaction design, except automatic completions, automatic suggestions, and follow-up requests. In addition, a subset of intentions is chosen to implement: asking for invoices (e.g., recently paid or of a certain value), revenue (e.g., in a certain period or averaged), costs, profit, and estimates (e.g., accepted or due soon). Invoices and estimates use the list modality, while key figures

use the rich modality that includes a table and plot. These intents are chosen because these intentions are common user actions in the application according to colleagues at Moneybird and are well represented in the small number of example requests that were collected among colleagues in an early phase (Appendix A).

Table 5.1: List of interaction design features and prototype inclusion status.

#	**Feature description**	**Implemented**
1	Show example requests before the user is typing.	Yes
2	Offer automatic completions while the user is typing.	No
3	Extract subphrases from a natural language request.	Yes
4	Map subphrases to natural language representations of concepts in a domain ontology.	Yes
5	Match mapped concepts to predefined slots and score intents.	Yes
6	Check if mapped slots fit previous request to adjust as critique.	No
7	Detect and communicate out-of-scope questions.	Yes
8	Detect and communicate ambiguities and contradictions in filled slots.	Yes
9	Compose a natural language search result summary reply.	Yes
10	Construct a query using the filled frame to retrieve results.	Yes
11	Suggest additional unused slots in natural language with the results.	No
12	Offer related search requests with the results.	No
13	Design and display different result visualizations.	Yes

5.2 Request Understanding

Understanding what the user intends is the first step in the prototype's process. This subprocess can be divided in two steps: request analysis, and intention understanding. This is also shown as the areas of blue methods in the architecture diagram in Figure 5.1. First, several NLP techniques are used to preprocess and analyse the request and tag its tokens. The request is divided into one or more subphrases using the request's dependency tree. This algorithm is explained in more detail further on. Second, these subphrases are mapped to concepts of designed Action and Object ontologies that represent bookkeeping in Moneybird. Each concept contains one or more Regular Expressions (REs) as its natural language representation to map to. The set of mapped ontology concepts that represents the natural language request is matched with the concepts defined for each intent. Intents

are scored and slots filled. The intent ranking and slot values are returned to the search component.

5.2.1 Request analysis

User input is first cleaned by removing most non-letter characters (e.g., punctuation). However, commas, hyphens, and underscores are preserved because of their grammatical role in a sentence or their common occurrence in named entities. All remaining characters are lowercased to remove case-sensitivity and ignore inconsistent use of capital letters in search queries [16]. There is one exception: the first letter of each word is capitalized as input to the named entity recognizer because empirical testing showed increased accuracy on selected sentences. The number digitizer detects written numbers and converts them to digits using REs to simplify temporal arithmetic. The request is then first analysed by tagging named entities, temporal expressions, money expressions, POS, and dependency labels on a token-level, and by parsing the dependency tree. Named entities and temporal and money expressions are tagged first. Multiple tokens belonging to a single expression or entity are merged into a single token to simplify the dependency tree.

POS tagging and dependency parsing are both achieved using the small Dutch pretrained model from *spaCy* [22]. The POS and dependency labels this model supports are described in Appendix B. NER is achieved using the standard four-class model for Dutch that comes with *Flair* [1, 44]. It was pretrained on the Dutch *CoNLL-03* dataset with *BERT* embeddings, reporting an F1-score of 92.58. Entities tagged as persons and organizations likely qualify for contact slots and are tagged as such.

Temporal expression tagging and normalization is implemented using REs and temporal arithmetic. Available implementations (e.g., *HeidelTime* [47] and *CoreNLP* [46]) lacked domain-specific temporal units such as fiscal quarters and were not compatible with the chosen development tools. The temporal expression tagger supports a range of expressions across a range of temporal units (days, weeks, month, fiscal quarters, and years), including absolute and relevant expressions, a time anchor, and ranges. Money expressions such as "*above 300 euros*" and "*between 10 and 50 euros*" are tagged similarly.

The dependency tree is also used to divide the request into one or more subphrases. The goal is to isolate individual pieces of information for slot value extraction. For example, the algorithm that was designed and implemented extracts the following subphrases from the Dutch sentence "*betaalde facturen van Moneybird die het afgelopen jaar verstuurd zijn*" ("*paid invoices from Moneybird that were sent last year*"):

1. "*betaalde facturen*" ("*paid invoices*")
2. "*van Moneybird*" ("*from Moneybird*")
3. "*die afgelopen jaar verstuurd*" ("*that sent last year*")

In this example, each subphrase now contains one piece of information. Subphrase 1 describes the state of the subject document, subphrase 2 the sender, and subphrase 3 when it was sent. The dependency tree of the sentence is shown in Figure 5.2. The expression "*afgelopen jaar*" ("*last year*") was merged into a single token because it was recognized as a temporal expression.

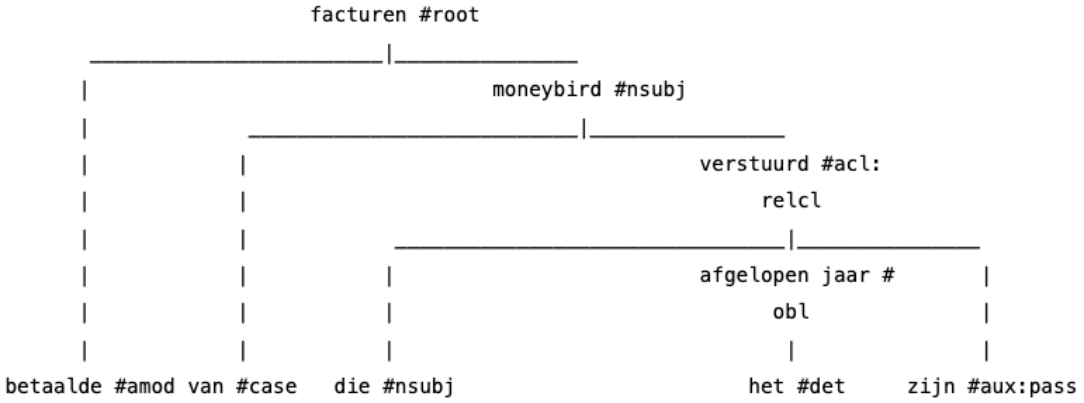

Figure 5.2: Dependency tree output of a Dutch search request.

The intuition is that a group of tokens depending on the same head forms an isolated part of a sentence. Therefore, the algorithm recursively traverses the tree, starting at its root, to find individual subtrees of tokens to form subphrases of. Which tokens are included depends on the type of subtree but tokens always keep their original order. A node can be a leaf or a subtree. Subtrees may be tall, deep, simple, or complex. These types of subtrees are visualized in Figure 5.3 and defined as follows:

- Tall: has only one child node.
- Deep: all nodes only have one child node until ending in a leave.
- Simple: has multiple child nodes, but is shallow (child nodes are all leaves) or at most one child node is a deep subtree.
- Complex: has multiple child nodes and is not a simple subtree.

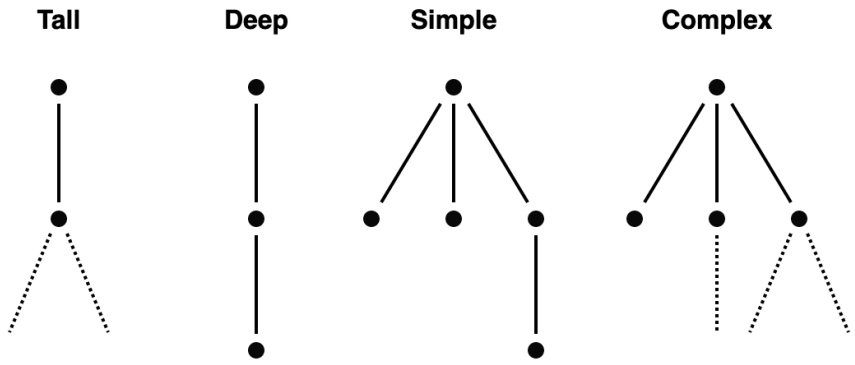

Figure 5.3: Visualization of subtree definitions used in the subphrase extraction algorithm.

Complex subtrees are evaluated recursively, while simple, tall and deep subtrees are resolved into subphrases. After resolving the subtrees of a complex node, its leaves and their head form a leftover subphrase. Pseudocode for the described algorithm is shown in Listing 5.1. It was designed empirically by examining example requests from Appendix A, their dependency trees, and the desired subphrases. Some tokens are left out of the subphrases based on their dependency label, such as determiners and auxiliary verbs. These tokens litter the subphrases because, despite them supporting the main verb, many sentences can still be understood without them. This helps keeping the REs simple. More examples are shown in Appendix C.

Listing 5.1: Pseudocode of the subphrase extraction algorithm.

```
function resolve(node):
    if node.is_leaf: return Subphrase(node.tokens)

    if node.is_tall:
        if node.is_deep:
            tokens = node.deep_tokens
            # include head of tree in the subphrase
            tokens += [node.head.token] if node.head
            return Subphrase(tokens)
        else:
            # resolve subtree at the end first ..
            subphrases = resolve(node.deepest_child)
            # .. before resolving this tall subtree
            tokens = node.deep_tokens excluding deepest_child

            # merge the tokens to the last subphrase
            # to prevent short leftover subphrases
            subphrases.last.merge(tokens)
            return subphrases
    else:
        if node.is_simple: return Subphrase(node.tokens)

        # complex, resolve non-leaves recursively
        subphrases = [resolve(child) for node.non_leaves]

        # add a subphrase from the leftovers (head + leaves)
        if node has leaves or all children were subtrees:
            tokens = [node.token + node.leaves.tokens]
            subphrases += Subphrase(tokens)
        return subphrases
```

Performance of the algorithm varies and strongly depends on the quality of the tagged dependency labels. Still, not every grammatically correct and correctly tagged request is divided optimally because of small or flat parse trees that are simply not subdivided. The results chapter also dedicates a section to the quality of this algorithm.

5.2.2 Intention understanding

Defining intents and slots

Intentions that are supported by the interaction are defined in an intent file consisting of a list of classification rules and a series of slots. Each frame is defined in a YAML file. YAML is a human-readable data serialization language that is commonly used for configuration files. An example of an intent definition file is shown in Listing 5.2. It describes the identifier and name of the intent, classification rules that reference two designed functions, and a list of slots. The named identifiers (e.g., `Bookkeeping.Period.DatePaid`) refer to concepts in either of two domain-specific ontologies that were designed according to the Moneybird application. A visualization of both ontologies can be found in Figure 5.4 below. One ontology represents the objects of the domain and their properties, and the other ontology describes the actions one can intend to undertake. The ontologies are not exhaustive as only concepts relevant to the scope of the prototype have been included. Classification rules and functions are discussed in a later section that describes scoring intents. An important distinction from a traditional frame-based architecture is that not all slots need to be filled to fulfil the request. Slots may act as optional filters. Some slots may be required or some may contradict each other if both are filled, but the system can clarify this during the interaction. Evaluating captured slot values occurs during response determination and is discussed in a later section.

Listing 5.2: Example YAML of the intent 'Find Invoices'.

```
---
id: 0
name: Find Invoices

classification:
- first_noun: Bookkeeping.Document.Invoice
- covers: Action.Retrieve.Items

slots:
- Bookkeeping.Document.Invoice.SalesOnly
- Bookkeeping.Document.Invoice.PurchaseOnly
- Bookkeeping.Document.Invoice.State
- Bookkeeping.TotalAmount
- Bookkeeping.Period.DateFinalized
- Bookkeeping.Period.DateSent
- Bookkeeping.Period.DateDue
- Bookkeeping.Period.DatePaid
- Bookkeeping.Contact.ByName
```

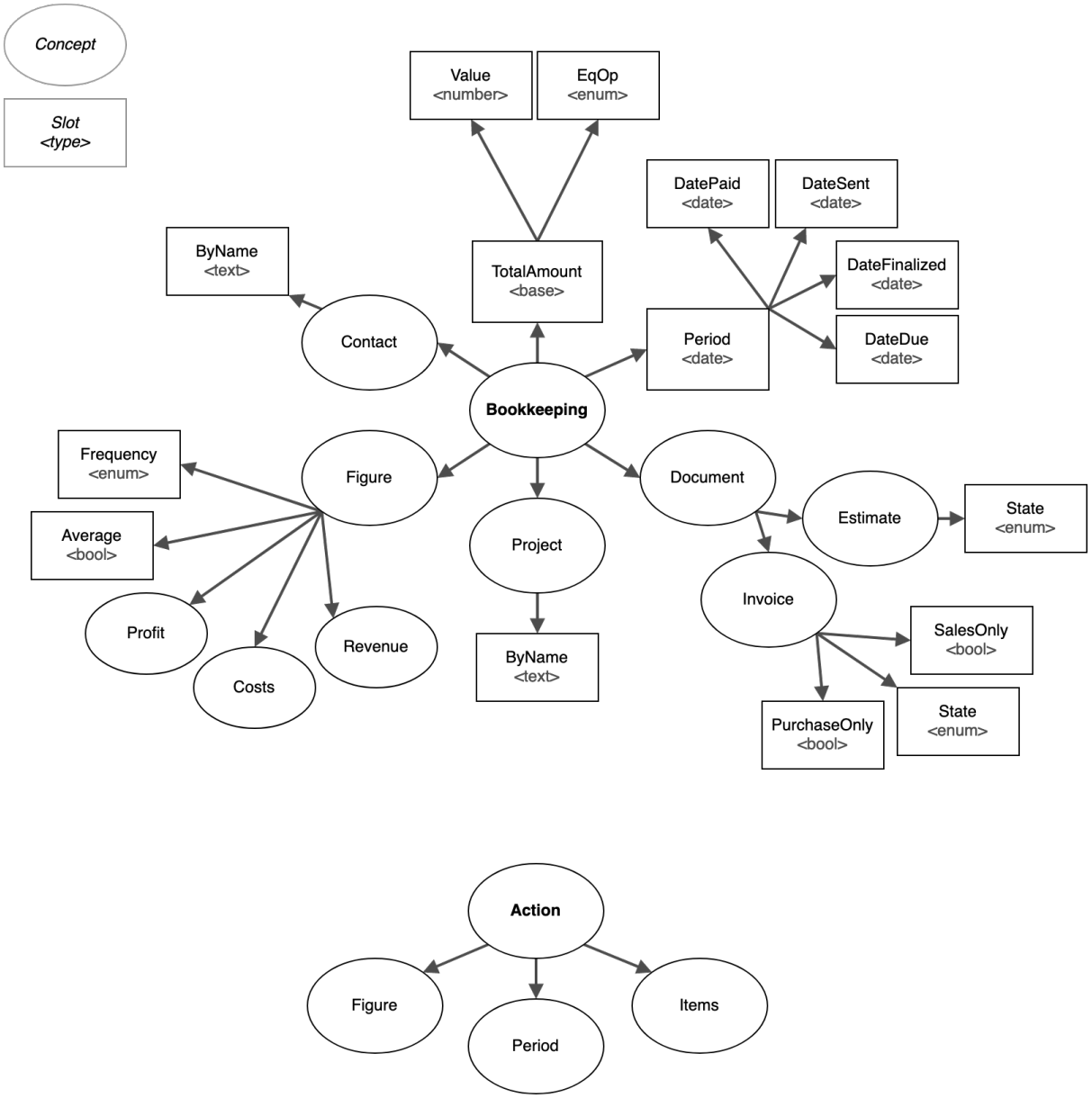

Figure 5.4: Diagrams of the designed Object and Action ontologies used in the prototype.

Modelling the ontology, mapping concepts, and filling slots

Each ontology is declared in code as a tree of concepts or slots. Slots are a more specific type of concept that may also capture information. Each concept (and therefore each slot) has a named identifier and type, and contains a list of REs—called triggers. Triggers model the natural language representations of the ontology concepts. For example, the concept `Bookkeeping.Document.Invoice` has one trigger: `/((ver|in)koop)?factu(ur|ren)/`. A subphrase is mapped to a concept if it matches any of its triggers. The REs allow for approximate matching (set to at most 1 insertion, or deletion) to handle minor spelling mistakes and typos. Mapping work by traversing all concepts from the root of either ontology for each subphrase. The set of mapped concepts is stored and used in scoring the predefined intents.

Slots are mapped too and also capture a slot value. A slot that is mapped extracts a slot value from the subphrase it is mapped to. Slot capture types include `base`, `text`, `number`, `boolean`, `date`, and `enum` values. A `base` slot consists of other slots that together make up its value. An `enum` declares a fixed set of keyed options that each have their own trigger. The slot value equals the key of the trigger that matches the subphrase the slot was mapped to. Slots of types `number` and `text` include an additional capture RE to fill them. A `boolean` type is considered true when the slot is mapped. A `date` type captures the normalized date range of the temporal expression that is tagged in the subphrase it was mapped to.

REs are used instead of word embeddings because preliminary testing with *FastText*'s Dutch static word embeddings[1] and *Flair*'s Dutch contextual word embeddings[2] did not show reliable cosine similarity scores between ontology concepts and related words. For example, "*open*" ("*open*") and "*onbetaald*" ("*unpaid*") have a similar meaning in the bookkeeping domain, but their word embeddings had low cosine similarity, while "*gesloten*" ("*closed*") and "*onbetaald*" ("*unpaid*") even had a slightly higher cosine similarity. Besides, REs are also useful to detect and extract specific numerical or textual information to capture slot values. Therefore, REs were chosen over word embeddings for the purpose of this prototype.

Classification functions and scoring intents

After ontology mapping, each intent is scored to assess how well it represents the user's search intention. A score is calculated using the classification rules stated in the intent definition file and the set of mapped concepts. Mapped concepts are all concepts of either ontology that were triggered by any subphrase of the search request. A classification rule states which classification function it uses and which ontology concept the function targets. The rule can optionally include a weight number to make some rules more or less important than others. Classification functions run on the entire request and not a particular subphrase. Three classification functions were devised for the prototype: `contains`, `covers`, and `first_<tag>`. These were empirically devised by comparing example requests collected among colleagues (Appendix A) and finding distinctive features. The reasoning behind these functions is that many search requests seem to concern an action and object (following the work of Lim and Lee [30]). For example, finding items, comparing date periods, and reporting on key figures. In many instances, example requests included mention of the object in the beginning of the sentence. Therefore, many requests could be distinguished if the first noun matches the intention's subject, such as

[1] `https://github.com/flairNLP/flair/blob/master/resources/docs/embeddings/CLASSIC_WORD_EMBEDDINGS.md`

[2] `https://github.com/flairNLP/flair/blob/master/resources/docs/embeddings/FLAIR_EMBEDDINGS.md`

invoices or revenue, and covered an action, such as asking for an amount or a set of items. The following list explains each function in more detail:

- `contains` checks if the target concept is included in the search request by testing the target concept's membership of the set of mapped concepts. It positively affects the score if the target concept was mapped, and negatively if it was not mapped.
- `covers` is similar to `contains`, but only increases an intents score if the target concept is included in the search request. It does not decrease the intents score if the target concept was not mapped. The function does not return a score directly, but increases the intent's *coverage* (a term discussed hereafter).
- `first_<tag>` checks if the first occurring token tagged with the specified POS or dependency tag would trigger the target concept. The accuracy of this feature depends on the quality of the POS and dependency tagger. For example, the classification rule "`first_noun: Bookkeeping.Document.Estimate`" checks if the first token that is tagged a noun matches `/offertes?/`.

Classification functions may return a score in the interval $[0, 1]$ (currently simply either 0.0 or 1.0). The weighted sum of these scores forms the base score for an intent. The base score is discounted using the intent's *coverage* to calculate the final score. Coverage is the ratio of the number of concepts that are matched to the intent to the number of all concepts mapped to either ontology. Concepts that are matched to the intent include concepts used in classification rules that resulted in non-zero scores, and mapped slots from the intent's slot list. This discount quantifies how well the intent definition fits the mapped search request. The final score decreases if not all mapped concepts also match the intent definition file. In other words, if the intent does not cover the search request well. An important consideration in calculating coverage is counting less specific ancestors of mapped concepts, because some concepts fully contain their descendants. This means a less specific ancestor would be mapped to the search request while only the more specific descendant is stated in the intent definition, causing an unfairly negative discount. For example, `Bookkeeping.Period` always triggers if `Bookkeeping.Period.DatePaid` triggers. The `Find Invoices` intent only lists the more specific slot and would not be able to achieve full coverage if `Bookkeeping.Period` were not counted too. The scoring function returns a number in the interval $[0, 1]$ and is shown in Equation 5.1. Each defined intent is sorted in descending order using its final score to form the ranking that is returned to the application by the HTTP API.

$$score = \left(\frac{\sum_{rule} score_{rule} \cdot weight_{rule}}{\sum_{rule} weight_{rule}} \right) \cdot \left(\frac{\#matched\ \ concepts}{\#mapped\ \ concepts} \right) \tag{5.1}$$

5.3 Response Determination & Query Building

The system uses the ranked list of intents, including scores and filled slots, to determine its response and build the relevant query to retrieve the results. Scores below a certain threshold indicate insufficient understanding and probably a request that is not (yet) supported. The prototype selects a frame based on the highest scoring intent, if its score is greater than or equal to a threshold value of 0.3. This value was empirically determined during development. Below the threshold value, the user's request is classified as not understood or supported. A high score indicates a high similarity between the definition of that intent and the interpretation of the user's intention. Before the system determines its final response, it checks the slot values for ambiguities or contradictions defined in the frame. For example, a user could request for invoices that were paid next week, but this is impossible because next week has not happened yet, and a user could request the average revenue, without specifying a certain time period. Ambiguity detection is done within the frame by providing boundaries (rules) about the (types of) values each slot may validly hold or other slots they contradict with. These boundaries are evaluated after the system selected the frame based on its score. If the system detects an ambiguity or contradiction, it uses this to determine its response. The reply generation part of the search component composes a proper natural language reply based on the determined response. Reply generation is discussed in the next section.

In case a frame was selected and no ambiguities or contradictions are detected, the generated natural language reflection of the system's interpretation is shown to the user. If the user selects the interpretation because it fits their intention correctly, the system retrieves the results by building a query for the RDB according to the frame's template. Frames build their queries by programmatically chaining ORM methods depending on which slots are filled. These methods can be used instead of writing raw SQL to query a database. Moneybird is built using Ruby on Rails, which uses the Active Record Query Interface[3]. Each slot is used to dynamically adjust a base query. Slots that are filled use their value to specify the query, while slots that are not filled do not adjust the base query. Depending on whether a certain slot is filled and its value(s), different methods are chained together to form and execute a query on the database. These methods include `where()` and `join()` to filter rows in a table and link data between tables, respectively. For example, if the slot `Bookkeeping.Period.DatePaid` is filled, the base query on the invoices table is adjusted by chaining `.where(paid_at: <date>)`, and if the slot `Bookkeeping.Document.Invoice.State` is filled with the value `unpaid`, the query is modified by chaining `.where(state: %w[open late reminded])`. In contrast, key figures such as revenue, costs, and profit are retrieved using an existing function in Moneybird, because this functionality was already present in the desired form for use on

[3] `https://guides.rubyonrails.org/active_record_querying.html`

the reports page within the application. The slots and their values are used to dynamically populate the arguments of the function, allowing the system to retrieve the desired results. Query building and execution on the RDB is deferred until an interpretation is selected, instead of when a search request is evaluated. This is necessary because the instant search feature may trigger multiple times while the request is still being typed, leading to a slower response time and increased strain on the Moneybird application. By delaying the query execution until a selection has been made, these negative impacts on the performance are avoided.

5.4 Answer Generation & Presentation

The system communicates a natural language reply on several occasions. When the user's request was not understood or supported, or if ambiguities or contradictions were found, to help the user understand what is wrong and how to improve their request. If a frame was selected, the system uses its template to dynamically formulate a reply that communicates its interpretation of the user's request. It reflects the intent and each filled slot in natural language by filling a string localization definition corresponding to the frame's template. When the user selects the interpretation to retrieve the results, the system also provides a similar dynamic natural language reply that acts as a result summary. This allows the user to quickly assess the results at a glance. This reply includes the number of items retrieved or the exact amount. These natural language replies are generated by the search component according to templates defined in the frames.

Dynamic reply generation is achieved using a rule- and template-based approach. Each frame provides its own reply template and natural language sentence components that can be dynamically combined based on which slots are filled and which are not. The data value of a slot is *humanized* into a natural language representation according to the slot's type. A frame's template provides the natural language strings that fit the generation rules of the slot type. An example of these string localization definitions that are used to construct a natural language reply for the `Find Invoices` frame can be found in Listing D.1 in Appendix D. Each frame must provide the same types of strings: reply templates in case of ambiguities, in case it is understood, in case zero results were found, in case one or more results were found, and strings per slot (defined according to the format of the slot type).

The frame template selects the correct base reply, depending on if it should return the interpretation of the request or the result summary. The template then stipulates which slots are required to complete the base reply. Slots can be assigned specific spots in the reply, while remaining slots are listed at the end. For example, the (Dutch) base reply for retrieved results for the `Find Invoices` frame is "*ik heb* `%{count}%{state}`

`%{invoice_type}` *gevonden*`%{appendix}`." The number of invoices found (`count`) and the type of invoices (sales, purchase, or both; `invoice_type`) are always completed, while the state filter (e.g., paid; `state`) and other slots (`appendix`) are only completed if the corresponding slots are filled. Optional parameters in the base reply are therefore not preceded by whitespace. The base reply's appendix is formed by listing the remaining filled slots in natural language. Similarly to Enlgish, in Dutch this means they are comma separated, while the last conjunction is "*en*" instead. The list below shows three example replies using this template and these rules:

- "*Ik heb 15 betaalde verkoop- en inkoopfacturen gevonden.*"
- "*Ik heb 5 verkoopfacturen gevonden die 'gemeente' als contact hebben.*"
- "*Ik heb 1 inkoopfacturen gevonden met een bedrag van meer dan € 100,00, die 'Adam' als contact hebben en die dit jaar toegevoegd zijn.*"

Replies to detected ambiguities and impossibilities are similarly implemented. For example, the keyed ambiguity `state_future_date` triggers if the user requests invoices that are already sent and of which the invoice date is in the future. The base reply for this ambiguity is "`%{state}` `%{invoice_type}` *kunnen geen factuurdatum in de toekomst hebben.*" and a completed example may be "*Betaalde verkoopfacturen kunnen geen factuurdatum in de toekomst hebben.*"

Slots are humanized before they are completed into a base reply. Each type has its own string localization definition requirements. Slots of type `text` and `number` simply insert their value into the defined string, unless the template defines a custom 'translator'. For example, a custom translator humanizes the `Bookkeeping.TotalAmount` slot by formatting the number as money first. Slots of type `enum` define a list of string localization definitions instead, keyed using the keys of the enum. Listing D.1 in Appendix D shows this for the `Bookkeeping.Document.Invoice.State` slot. Slots of type `bool` use their value to insert the defined string in the base reply. For example, when the average is requested for a key figure. Slots of type `date` humanize their value according to several rules to provide a specific translation that is not as verbose as an absolute date and time string. Table 5.2 below shows examples of each variant of date value humanization.

The UI is implemented using traditional front-end technologies (HTML, CSS, and JavaScript), and by repurposing existing code from the application. The following figures show the UI of the developed prototype in several situations during the interaction. Figure 5.5 shows the idle search box that includes a placeholder suggestion. Direct interpretations repeat the filled slots in the natural language replies, as shown in Figures 5.6 and 5.9. Interactive feedback tells the user what is ambiguous about their request, as shown in Figures 5.8 and 5.11. The system shows the results to the user—if any—according to the result modality. Documents are shown in a list, while key figures are states together with a table and plot, as shown in Figures 5.7 and 5.10, respectively.

Table 5.2: Examples of date humanization per variant.

Date range value	**Natural language string**		**Variant**
`20230131-20230131`	"*vandaag*"	("*today*")	relative
`20230130-20230205`	"*deze week*"	("*this week*")	relative
`20220101-20221231`	"*het afgelopen jaar*"	("*last year*")	relative
`20221212-20221212`	"*op 12 december 2022*"	("*on 12 December 2022*")	one day
`20220701-20220930`	"*in K3 2022*"	("*in Q3 2022*")	period
`20210201-20210228`	"*in feb 2021*"	("*in Feb 2021*")	period
`20190101-20191231`	"*in 2019*"	("*in 2019*")	period
`20221201-20230131`	"*tussen 1 december 2022 en vandaag*"	("*between 1 December 2022 and today*")	between

Figure 5.5: Dutch UI (cropped) of the prototype search functionality implemented in Moneybird; displaying a focused placeholder suggestion underneath the search box.

Figure 5.6: Dutch UI (cropped) of the prototype search functionality implemented in Moneybird; displaying a typed request and the focused interactive feedback underneath acting as the direct interpretation.

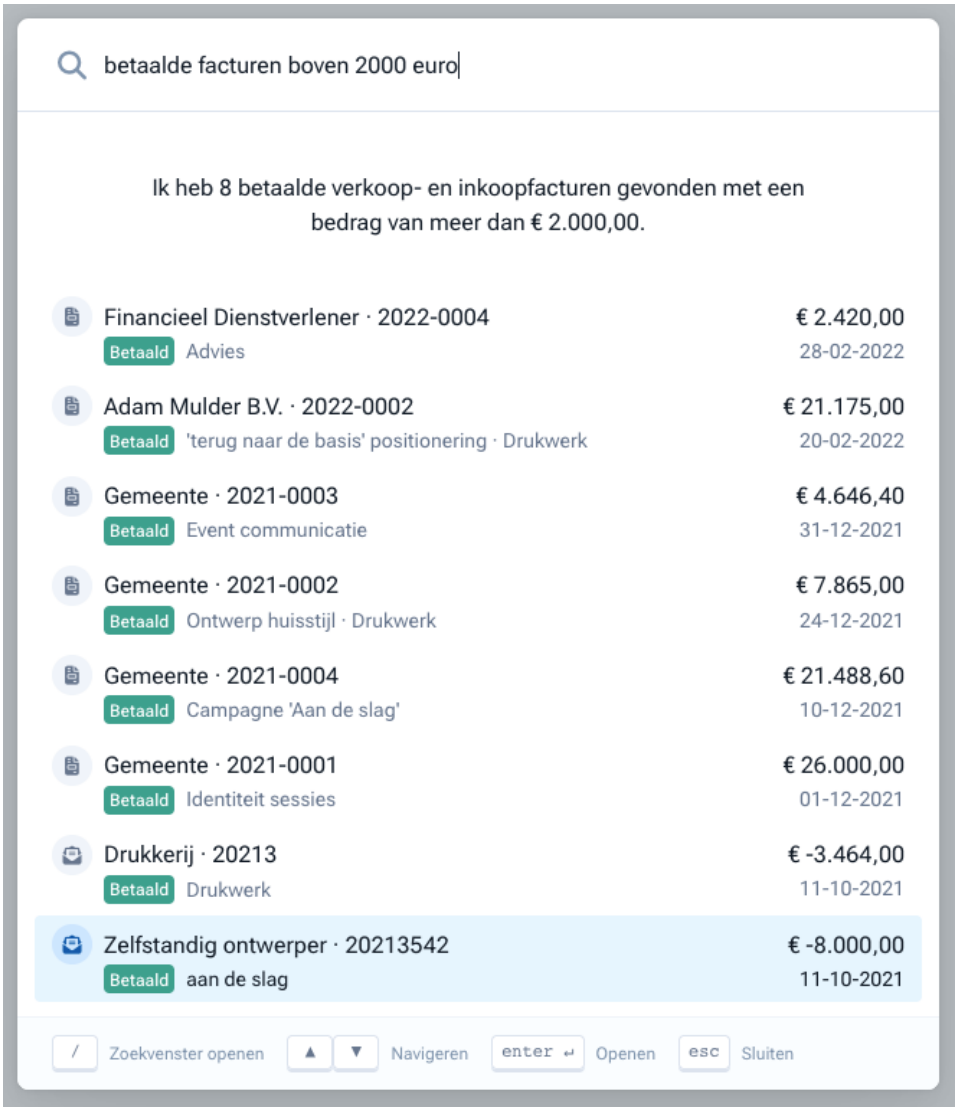

Figure 5.7: Dutch UI (cropped) of the prototype search functionality implemented in Moneybird; displaying paid invoices with a minimum amount using the List modality.

Figure 5.8: Dutch UI (cropped) of the prototype search functionality implemented in Moneybird; displaying a typed request that includes an impossibility and the focused interactive feedback underneath.

Figure 5.9: Dutch UI (cropped) of the prototype search functionality implemented in Moneybird; displaying a typed request and the focused interactive feedback underneath acting as the direct interpretation.

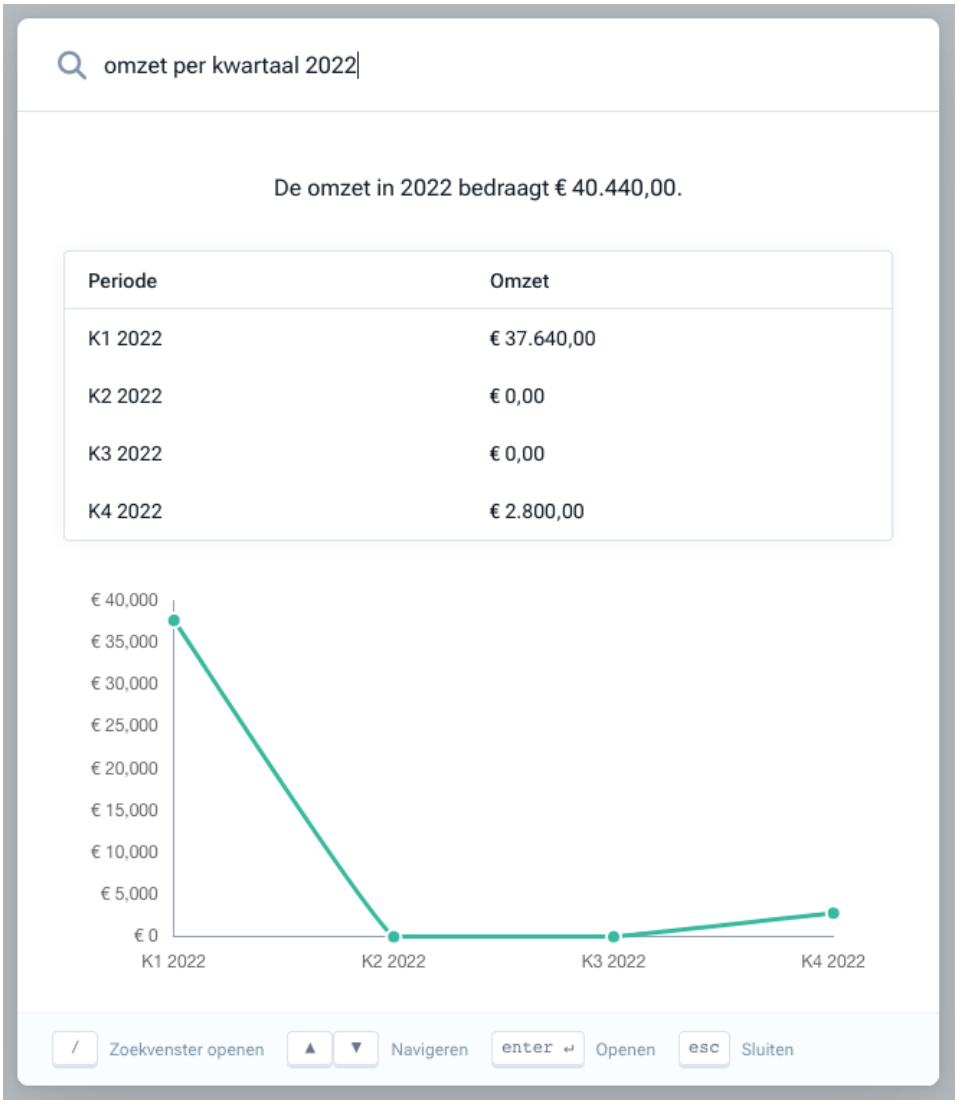

Figure 5.10: Dutch UI (cropped) of the prototype search functionality implemented in Moneybird; displaying the revenue of 2022 per quarter using the Rich modality.

Figure 5.11: Dutch UI (cropped) of the prototype search functionality implemented in Moneybird; displaying a typed request that is underspecified and the focused interactive feedback underneath.

6 User Study

The prototype of the conversational IR interaction is evaluated using a user study to research how real users experience the system's helpfulness and limitations in relevant scenarios. This is especially important because faster or more accurate systems are not always perceived as 'better' [27]. This chapter details the study's setup, protocol, participants, and data collection and analysis method. It is based on how interactions with task-oriented agents and other intelligent systems are commonly evaluated in user studies, as reviewed in the previous phase of this master's thesis [45, p. 24].

6.1 Protocol & setup

The user study consists of participants conducting a series of search scenarios; each using the application's traditional navigation and keyword search functionalities, and the prototype (within-subject design). They are instructed to think-aloud while conducting the tasks, and they fill in a questionnaire about their experiences after completing each scenario. Before the first scenario takes place, participants are asked about how they would expect the interaction to work. After the scenarios, a discussion takes place about the participant's experience and how it related to their initial expectations. They are also asked to think of use cases where this interaction would be really useful during their day-to-day dealings with the bookkeeping software, or may be not useful at all. The user study is conducted at the Moneybird office. Within the office, a meeting room was prepared with a table, two chairs, and a laptop; mimicking a comfortable environment for the participant. After welcoming the participant, they are briefed in this room and sign the consent form if they have not done so yet during recruitment. The researcher is present during the experiment to take notes and instruct the participant. Audio is recorded using the computer the study is conducted on to capture any comments the participants may make. It is made clear that the system is tested and not them. There are no right or wrong answers to their natural language input. After the session, they are debriefed and any questions they may still have are answered. They are thanked and provided with a Moneybird-standard incentive (€50,- gift card and €0,19/km compensation for travel expenses). The whole procedure lasts a maximum of one hour and thirty minutes.

A diagram of the above procedure is shown in Figure 6.1 and a copy of the instructor protocol can be found in Appendix G.

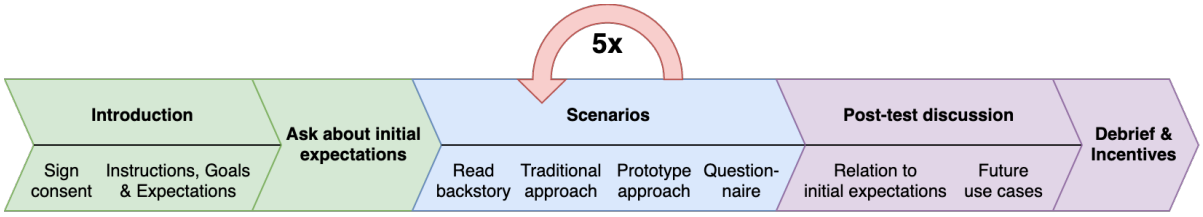

Figure 6.1: Diagram of the protocol of a user study session.

Participants go through five search scenarios in an artificial financial administration. All scenarios are possible to complete in both the current system and the prototype interaction. The described information need in each scenario increases in complexity. Each scenario is introduced to the participant by means of a written backstory to allow participants to formulate their own request and to prevent influencing request formulation [49] by distributing the separate details of the information need over the backstory. A copy of the backstories that were provided to the participants are found in Appendix H. The five search tasks are described below:

1. Finding purchase invoices that have to be paid.
2. Finding the costs of the first quarter of this financial year.
3. Finding an estimate that has not yet been accepted with an amount between €100 and €150.
4. Finding sales invoices that were sent to the municipality last year.
5. Finding the average revenue per quarter of the 'consultancy' project in 2021.

These tasks were chosen because these scenarios cover the different intents and types of slots that were implemented in the prototype, while getting increasingly complex by including more information in the backstory that needs to be included in the request. As an example, the English version of the backstory of scenario 5 is written below. Details of the information need are shown in bold to clarify the distribution over the story for the reader. This is not present in the handouts.

> **Last year**, you provided advice to several clients by means of interactive online workshops. You are thinking about giving these workshops again to new clients in the coming year. At the time, you booked the **revenue** under **a project called 'consultancy'**. To consider this, you want to know how much you earned then **on average** and **per fiscal quarter**. Look up this number.

Participants first use the traditional navigation and search functionalities to complete each scenario to help them understand the scenario's goal and allow them to compare

their experiences more concretely. The two approaches are not more directly compared (for example, by assessing time to task), because it is more valuable to find out *why* one approach is preferred over another and because the results of a direct comparison are highly dependent on the design of the context application.

A pilot study was conducted to prepare the researcher, test the location and tools, and to improve the clarity of the instructions and backstories. Two colleagues participated in the pilot study one week before the user study. This led to improvements to the written backstories and to helpful notes on the protocol, but no changes to the design were necessary. Ethical concerns of this study design were evaluated and approved by the Ethics Committee - Computer & Information Science of University of Twente[1].

6.2 Participants

Conducting a user study is costly and time-consuming, which limits the number of participants that can be incorporated in the research compared to lower-cost studies, such as questionnaires and diary studies [4]. Based on a review of evaluations of similar intelligent systems, the number of participants varies greatly [45, p. 24]. However, 15 to 40 participants is common. Considering the available time, the goal is to conduct sessions with around 25 participants.

Inclusion and exclusion criteria for participants promote a generalizable sample and aim to protect the study from invalid samples or likely outliers. Inclusion criteria for this study are that participants be Dutch-speaking and clients of Moneybird. Clients that have been using the bookkeeping software for less than one year are excluded to increase the likelihood of them having a sufficient understanding of the bookkeeping software. This is important because the bookkeeping software is merely the context of the user study, and a lack of understanding may hinder the interaction. Minors are excluded to prevent complications with participation and consent.

Participants were recruited by publishing a signup form within the application's research tool. This tool is often used by Moneybird to collect customer insights. The form clearly states the details of the study and what is expected of them to prevent surprises on the consent form or when they are on location. A copy of the signup form can be found in Appendix E; and a copy of the consent form can be found in Appendix F. The signup form's questions test inclusion and exclusion criteria, frequency of use of smart assistants, and demographics to allow for a valid and representable selection of respondents to be invited. 134 clients responded to the recruitment form and answered the questions. The results of a frequency analysis of their answers is shown per variable in Table 6.1.

[1]Reference number RP 2022-175

Table 6.1: Frequency analysis of answers to the recruitment form concerning frequency of use of smart assistants (top left), age group (top right), prior experience with Moneybird (bottom left), and travel distance (bottom right).

Answer	Freq.	Percent
Never	33	24.6%
Almost never	32	23.9%
At least once a month	16	11.9%
At least once a week	21	15.7%
Daily	32	23.9%

Answer	Frequency	Percent
18–29 years	28	20.9%
30–39 years	41	30.6%
40–49 years	26	19.4%
50–59 years	26	19.4%
60+ years	13	9.7%

Answer	Frequency	Percent
0–0.5 years	17	12.7%
0.5–1 years	15	11.2%
1–5 years	63	47.0%
5+ years	39	29.1%

Answer	Frequency	Percent
0–25 km	18	13.4%
25–65 km	13	9.7%
66–100 km	22	16.4%
100+ km	81	60.4%

The results of both the age group and frequency of use of smart assistants variables happen to be desirably varied across the answer options. Prior experience of Moneybird shows that the vast majority of respondents have more than one year of experience with the bookkeeping software. Travel distance was important because it may be indicative of how likely respondents are to follow through with participating in the study.

Conducting 134 sessions was deemed unfeasible and too expensive regarding the financial incentive. Only participants that have one or more years of experience with the bookkeeping software and indicated a travel distance of less than 100 kilometres were selected, resulting in a selection of 48 participants. This number is still almost double the aim of recruiting 25 participants. Over-recruitment compensates for people that misunderstood the recruitment form, cannot find the time to schedule a session, do not respond any more, do not show up, or forget, or situations where something has simply intervened. The 48 selected participants were invited on 5 October 2022 to book a session between November 1 and November 18, 2022. The invitation included information about the financial incentive, a detailed description of the activities, and a copy of the consent form to present surprises on location. A reminder was sent on 17 October 2022 to each invited participant that had not yet booked a session yet.

6.3 Data collection & analysis

To find out how the natural language search interaction helps or limits participants during search, participants think-aloud and the researcher discusses his observations and their experiences with them. Furthermore, their opinion is quantified across 12 measures. As discussed in the beginning of this chapter and visualized in Figure 6.1, after the introduction and before showing the prototype, the participant is asked about their initial

expectations of how the interaction works. Afterwards, they are asked how their experiences aligned with their initial expectations and in what use cases in the application this interaction would benefit them or would not be helpful. During the sessions, the requests the participants type are automatically collected.

The 12 measures collected using the post-scenario questionnaire are based on questions from several related usability studies [52, 53, 48, 15] (expressed as 5-point Likert scales) and Brooke's System Usability Scale [5]. Five-point Likert scale questions cover how participants perceive expected behaviour, ease of use, cognitive effort, capability clarity, understanding quality, response clarity, result correctness, interaction pace, response speed, learnability, satisfaction, and future use. The questionnaire is filled in after each scenario to measure change while the participant gets more experienced with the prototype. It is made clear to them that they should answer the questions by quantifying their entire experience up to this point, instead of the experience of the particular scenario in isolation. A copy of the survey can be found in Appendix I. Table 6.2 details the full list of measures used to evaluate the prototype interaction and how they are collected. Common IR evaluation measures (e.g., precision, recall, and F measure [33, p. 155]) are excluded because the goal is to evaluate how participants experience the natural language search interaction and not to find the most relevant results for a specific request. Inaccurate results during the user study sessions are considered by evaluating how accurate participants perceived them.

The audio and notes are analysed by performing thematic analysis [7]. Relevant comments are coded to analyse common themes. Codes are aptly named and the number of participants codes occur in, is counted. Themes are devised based on sets of related codes. Some codes may be discarded because they are too vague or not relevant enough. Themes are reviewed to make sure they are useful and represent the original data well. Results consist of a list of named themes with detailed definitions describing them. These themes thus describe what helped or limited the participants with resolving their information need while using a natural language search interaction. Relevant quotes are also collected from the audio to strengthen themes by showing concrete examples. The entire process is visualized in Figure 6.2.

Figure 6.2: Diagram of the thematic analysis process.

The quantitative data collected using the post-scenario questionnaire is analysed per variable per scenario to discuss potential change in these variables while participants get more experienced with this alternative way of searching. The variables are not statistically analysed per group of the age and frequency of use of smart assistants variables, because

Table 6.2: Overview of quantitative evaluation measures collected during the user study

Measure	**Description and collection method**
Turns needed	Number of requests submitted to complete the task. Search requests are recorded and counted.
Expected behaviour	Perception of how much the interaction works as the participant expects. Likert scale in the post-scenario questionnaires.
Ease of use	Perception of how easy it was to use the prototype. Likert scale in the post-scenario questionnaires.
Cognitive effort	Perception of how much cognitive effort it took to use the prototype. Likert scale in the post-scenario questionnaires.
Capability clarity	Perception of how clearly the prototype communicated its capabilities. Likert scale in the post-scenario questionnaires.
Understanding	Perception of how well the participant's requests were understood. Likert scale in the post-scenario questionnaires.
Response clarity	Perception of how well the participant understood the responses. Likert scale in the post-scenario questionnaires.
Result correctness	Perception of how accurate the prototype was. Likert scale in the post-scenario questionnaires.
Interaction pace	Perception of how comfortable the interaction's pace was. Likert scale in the post-scenario questionnaires.
Response speed	Perception of how fast the prototype responded to requests. Likert scale in the post-scenario questionnaires.
Learnability	Perception of how quickly the participant became familiar with the interaction. Likert scale in the post-scenario questionnaires.
Satisfaction	How satisfied the participant was with using the interaction. Likert scale in the post-scenario questionnaires.
Future use	If participants would want to use the prototype again in the future. Likert scale in the post-scenario questionnaires.

the sample size assumption of Analysis of Variance (ANOVA) cannot be satisfied with the aimed number of participants. ANOVA would be the proper statistical method for this use case, but the assumption stipulates to have at least 30 data points per group, which would require a minimum of 150 participants. The collected search request logs are exported as Comma Seperated Values (CSV) and analysed in Apple's Numbers (Microsoft's Excel for macOS). Logs include uncompleted queries because the search functionality features instant search. This means the search is performed when the user stops typing (or pauses for a moment while pondering their request). These lines are manually removed. The number of turns is then based on the resulting number of lines per scenario that agrees with the number noted during the session.

The performance of the prototype system is also evaluated to place the user's experiences in perspective. The performance is evaluated using the logged search requests that the participants produce during the user study. Only the requests that are unique after the prototype's preprocessing step are considered, because the system is deterministic and identical requests will yield identical intent determination and slot filling results. Each request in the CSV file is manually labelled to include the scenario it concerns, and if the request more resembles a keyword query (e.g., "*gemeente 2021 verstuurde facturen*") or a natural language request. This distinction is made to provide more insight into many of the performance metrics, and determined by the author. The file further contains the API responses in JavaScript Object Notation (JSON) that contain the intent ranking and filled slots. Evaluation covers the subprocesses within the request understanding component: POS tagging, dependency tagging, NER, temporal expression tagging, money expression tagging, subphrase extraction, ontology mapping, slot filling, and intent scoring. The two pretrained models used in the prototype system (for POS and dependency tagging, and NER) are not evaluated by the author, but metrics reported by the authors of the models are discussed instead. Temporal and money expression tagging are evaluated by manually counting how often expressions in the unique logged search requests are correctly tagged. Subphrase extraction is evaluated by performing a frequency analysis on the number of subphrases requests are divided in, manually counting how many divisions are undesired, and discussing examples. Subphrases are undesired if they split up words that must stay together (e.g., verb and its subject) or one contains a similar type of slots more than once (e.g., multiple temporal expressions). Subphrases are desired if they are not undesired. Ontology mapping is evaluated by calculating the intent ranking accuracy—how often the right intent is scored highest (above the threshold)—using the added scenario numbers and the JSON response and discussing examples. The intent ranking accuracy is calculated across scenarios, per scenario, and for keyword queries and natural language requests. Slot filling is evaluated by calculating the Slot Error Rate (SER) [25, p. 548]. This is the number of slots that are not filled (with the correct value), divided by the total number of slots in the logged search request. These numbers are both manually counted in the CSV file. The SER is also calculated across scenarios, per scenario, and for keyword queries and natural language requests. Finally, the task completion rate is calculated to assess the ratio of search requests that are entirely correctly understood, by finding the unique logged search requests that scored the correct intent the highest (above the threshold) and have an SER of 0.00. The task completion rate is also calculated across scenarios, per scenario, and for keyword queries and natural language requests.

7 Results & Discussion

7.1 Participants

Nineteen user study sessions were conducted between November 1 and November 24, 2022. Twenty-three out of 48 invited participants eventually booked a session. This means 25 out of 48 (52.1%) were unable to schedule an appointment. Less than half responded with a cancellation, while 13 invited participants did not respond to the invitation nor reminder. During the user study, three participants that had booked a session had to cancel due to circumstances and one participant did not show up. Despite the lower turnout than aimed, the over-recruitment strategy proved useful. Frequencies of the characteristics of participants' age group and frequency of use of smart assistants for the 19 participants are shown in Table 7.1. Although participants still show characteristics that vary across variable groups, it is less varied than the original selection of 48 invited participants. All invited participants that never use smart assistants did not end up participating, and the 40–49 years age group showed the largest decrease (−17.6 percentage points), with only one remaining participant.

Table 7.1: Frequency analysis of answers to the recruitment form of the 19 participants concerning frequency of use of smart assistants (left) and age group (right).

Answer	Freq.	Percent
Never	0	0.0%
Almost never	4	21.1%
At least once a month	4	21.1%
At least once a week	4	21.1%
Daily	7	36.8%

Answer	Frequency	Percent
18–29 years	6	31.6%
30–39 years	7	36.8%
40–49 years	1	5.3%
50–59 years	3	15.8%
60+ years	2	10.5%

7.2 Thematic Analysis

Eighteen codes were identified during the first phase of thematic analysis. These were named and counted, and are shown in Table 7.2 below. Related sets of codes formed four major themes about natural language search and how it helped or limited participants with their information needs in bookkeeping software compared to traditional search.

The following sections detail an elaborated description of each identified theme. The themes were named and are listed below:

- Trust and initial expectations
- The path of least resistance
- Learning the system and breaking barriers
- Naturalness

Table 7.2: Overview of 18 codes identified using thematic analysis and the number of participants these occurred in.

Identified code	**Number of occurrences**
Initial Expectations	14
Benefits of Keywords	14
Input Uncertainty	13
(Completion) Suggestions	12
Quality Affecting Use	11
Less Effort than Navigating/Filtering	11
Learning the System	11
Aiding Awareness	11
Interactive Guidance	11
Speech	10
Thought-driven Natural Language	9
Natural Language Support Disbelief	8
Trust	7
Expert Users	7
Describing Uncertain (Paths to) Information Needs	7
Describing Complex and Specific Information Needs	7
More Effort than Navigating/Filtering	6
Human Likeness	5

7.2.1 Trust and initial expectations

The following five relevant codes were related to form the theme "Trust and initial expectations": *Initial Expectations*, *Input Uncertainty*, *Quality Affecting Use*, *Natural Language Support Disbelief*, and *Trust*.

The use of a natural language search interaction is massively impacted by the amount of (initial) trust of its users and their initial expectations. Having trust in how freely they may formulate their natural language request; how well their request will be understood (as also found in [13]); and the correctness of the results all affect the interaction. Their trust is affected by their initial expectations of the interaction and the system's capabilities and qualities, and their trust can be built or degraded over time with use of the system. The performance and quality of the system positively and negatively affects the users' trust and therefore their use of the system. This is consistent with previous findings by Yu *et al.* [58].

The initial level of trust is set by users' initial expectations. The user study showed again what is known: people are conditioned to using keywords. This (subconsciously) affects their use of a natural language search interaction. Search systems in general do not utilize NLU and users are therefore hesitant to try, despite introduction and hints provided by features of the interaction. A lack of confidence in the system's capabilities to understand their natural language input limits participants and has them entering keywords. This feeling is influenced by previous experiences—especially negative ones—with systems utilizing natural language. For example, chatbots. Participants mention understanding that these often trigger on specific words, and therefore intended to game this system in the same way. When users have little experience with a system's capabilities and do not (yet) trust their natural language use will be understood correctly, they show significant hesitation during request formulation. They reveal feeling the need to know what language the system accepts and in what format it is required. Users think they must find a suitable question to ask by adhering to a certain request format or targeting specific words to make it work. This is consistent with previous research [27, 13] and has users unnecessarily think about their natural language use and increases the required cognitive effort during request formulation. Degraded NLU quality of the system increases this feeling. It makes people lose their trust in being able to type their need freely, undermining a benefit of natural language.

Not all users are sceptical because of negative experiences with similar systems. Others have especially positive experiences with interactions using natural language or have no particular expectations and trust the system will do what it is introduced to do. Some of these participants formulated more proper natural language requests. However, many participants that had their initial expectations aligned with the system's form of input did not formulate natural language requests from the beginning. They hardly did

so because they felt uncertain about this system's capabilities, despite understanding what kind of input it expects. This lack of trust and the (subconscious) conditioning to keywords makes participants more cautious about entering a proper natural language request. Some participants mentioned keywords should still be understood despite being a natural language search system. How well the system performs affects these levels of trust. Bad performance degrades the level of trust in the system's level of request understanding. It has people showing greater hesitation in subsequent request formulation and even falling back to entering keywords. Good performance in request understanding builds that trust. It also has users gain the confidence to try and seek the boundaries of the system's capabilities. With this trust in understanding ability, users (subconsciously) try both more free-form requests with less worry, and more keyword-style requests to save time and effort.

7.2.2 The path of least resistance

The following six relevant codes were related to form the theme "The path of least resistance": *Benefits of Keywords*, *Less Effort than Navigating/Filtering*, *Expert Users*, *Describing Uncertain (Paths to) Information Needs*, *Describing Complex and Specific Information Needs*, and *More Effort than Navigating/Filtering*.

Users want to resolve their information need as quickly and effortlessly as possible and have multiple ways of finding what they need: natural language search, keyword search, and navigating plus filtering. Which path to take is not an absolute consideration dictated by the application's design, but much a personal one, depending on many factors. The total time spent searching is not the only critical aspect in this equation. The cognitive effort needed and the physical effort needed are especially important. These relate to the complexity of the interaction and the amount of action the user needs to undertake, respectively. These aspects are influenced by the application's design, but personal factors also play a role. Users choose the path of least resistance for each information need.

Keywords are faster to type, and only some users experience more difficulty selecting them, because most users are used to working with keywords for years and have trained to select proper terms. Despite natural language requests being more intensive to type, participants mentioned several reasons why it is sometimes worth doing so. First, expressing your information need in natural language helps to describe domain specific language, application terminology or jargon (e.g. asking for "how much tax do I get back?" instead of "recoverable VAT"). This can be language people are unfamiliar with or that is difficult to remember. Second, complex navigation and filtering is expressed in natural language. Describing the navigational destination and its filtering in a sentence is experienced as taking less effort than navigating to the relevant web page, interpreting the filter UI, and selecting the correct options. Filter options can also be described collectively or using negated terms. For example, asking for "unpaid invoices" instead of selecting two invoice

state options "open" and "late". However, this depends on the relative complexity of the information need and the experience of the user with the application. Third, specific or *deep* information needs are expressed in natural language by describing a document by its details, such as contact, state, and date. Very specific information may be located in deeper parts of the application, and it typically requires more navigational actions and filtering to uncover. Participants mention the importance of this tactic in environments where more data is stored (compared to the relatively empty test administration). Last, users use natural language to express information that they are unsure exists, or do not know via what route to find. Describing the information they are looking for avoids searching for a path through the navigation they may be unaware of or that may not even exist. Expert users set a high bar for search functionality because they know their way around their data. This raises the threshold of typing a natural language request and decreases the likelihood that this type of users adopt natural language search, because they are probably faster 'the usual way'.

Choosing to resolve an information need using a natural language search interaction is a personal consideration, made again for each individual search case. It involves the complexity of the information need, the amount of action needed with either approach, and the speed of performing either approach. These three factors are influenced by application design and personal factors, but also affect each other. The cognitive effort involves actions that users need to think about, such as navigating the UI, scanning filter forms and selecting relevant options, selecting keywords, or formulating a natural language request. Navigating the UI and using filter forms is less intensive for expert users and if the information is not located deep in the application. Selecting keywords is less intensive for people that have affinity with digital technology or are conditioned and trained to use keywords through experience. The effort of searching using natural language is affected by trust and initial expectations, and the performance of the system. The physical effort involves the amount of action to take using the UI or the amount of typing to perform. The first is influenced by the application's design and the depth of the information needed. The latter by the detail required to express the information need and the user's belief of the expected request formulation criteria. The speed involves the time it takes to resolve the information need. Typing keywords is faster than an entire sentence, depending on the cognitive effort it takes to formulate either. The speed of typing a search request versus navigating and filtering the UI is influenced by the depth of the information needed, the detail required to express the information need, being an expert user, and the typing ability of the user. The effort of writing a natural language request can weigh up to the traditional approaches, but that depends on the relative effort of these approaches, which can be partly influenced by better application design. This all makes searching in natural language a personal consideration that is made again for each individual information need.

7.2.3 Learning the system and breaking barriers

The five following relevant codes were related to form the theme "Learning the system and breaking barriers": *Benefits of Keywords*, *(Completion) Suggestions*, *Learning the System*, *Aiding Awareness*, and *Interactive Guidance*.

> *"A search program is really only with keywords. That's how Google works, that's how everything works."* – P20

Conditioning to using keywords is strong because of decades of use and experience with search systems. Extra words that help form a grammatical sentence and connect context (e.g., determiners, adpositions, adverbs) traditionally litter a keyword search by introducing irrelevant results because they appear very frequently or carry less specific meaning than most verbs and (proper) nouns. This conditioning is a high hurdle for an interaction that uses natural language to search. This feeling is strong enough that simple capability hints, such as placeholder suggestions, are overruled. Participants mention reading request suggestions before formulating their own request, but not believing it will really work that way.

> *"Adding all these words doesn't make me think: now I will find it!"* – P11

Participants needed to learn how this natural language search interaction worked, because it does not function as many similar search systems do and has an unfamiliar UI. While getting more experienced with the interaction, they learned how to enter requests, and how to interpret the interactive feedback and results. Many participants figured out the function of these different components after interacting with them multiple times. Although the design and goals of the UI, the request suggestions, and the interactive feedback were not immediately clear to all participants, it was more straightforward than learning (and trusting to learn) what kinds of requests the system expects from the user. After several interactions, most participants learned that the interactive feedback showed them what was understood and what was missing, causing them to adjust their request without looking at the results or having confidence that the results would be what they were looking for. However, not everyone expected every aspect of their request to be reflected, while others thought everything should be reflected and were confused by interactive feedback that did not understand every part of their sentence.

Searching is a process of going back and forth between request formulation and assessing the results [2, 19]. Search systems interactively help the user towards their answer by providing completion suggestions and related search queries. Participants expect these features in a search function because they are used to them in similar systems. Despite completion suggestions not being implemented in the prototype, participants mentioned the feature, and that using completion suggestions is quicker than typing and helps them formulate their request when not sure what they are looking for or what words to use.

One participant did not complete their request because nothing happened after starting it, and therefore was not sure that what they wanted to type was supported by the system. Request suggestions aid awareness of system capabilities: what is possible to ask and how requests may be formulated. However, this awareness is not enough to persuade people to search using natural language. Breaking the barriers of conditioning involves being 'rewarded' for natural language use by being correctly understood and finding the right results.

Breaking the barrier of conditioning involved several other themes identified by the thematic analysis. Trust and initial expectations and naturalness play a big role in eliciting natural language use.

7.2.4 Naturalness

The three following relevant codes were related to form the theme "Naturalness": *Speech*, *Thought-driven Natural Language*, and *Human Likeness*.

When participants think about their information need, the first thought that comes to mind is often expressed in natural language. However, many participants have learned and are conditioned to immediately transform this need into a small set of keywords. Even subconsciously. Only some participants explicitly mention that formulating a natural language expression of their information need is less cognitively intensive than selecting a set of keywords; allowing them to skip a step in the conditioned thought process. The following quotes show this naturalness:

> *"I literally type what I am looking for."* – P12
>
> *"You simply ask a question. The sentence that comes to mind."* – P16
>
> *"It is useful you can describe it the way you normally think."* – P7

However, many participants are trained to use keywords well, and some mention them being the first things that come to mind when reading the search scenario.

Speech elicits natural language use and has established use in voice-enabled smart assistants. Participants understand or are used to the fact that this does not work with keywords, and one participant mentions using keywords with a voice-enabled smart assistant would lack context. Many participants mentioned imagining using a conversational search interaction vocally without being asked about it. Major reasons are typing being slower than speaking and taking more physical effort.

An interaction that feels like talking to a person elicits natural language use, despite the user typing their request. This is not the case if the interaction feels like working with a computer or machine, unless its goals are crystal clear (e.g., translation software). Only a few participants explicitly mentioned feeling like talking to a person when entering search requests, while many mention the natural language search interaction lacked humanity;

humanity that may help elicit natural language use. One participant mentioned that the first-person singular use of "*ik*" ("*I*") in the result summary made him feel more likely to input a natural language request.

> *"I know how to ask the question face-to-face, but when asking it to a screen, then I don't."* – P18

7.3 Quantitative Analysis

The frequency of each number of turns needed to complete a scenario is found in Table 7.3. Participants' first interaction with the system is a new experience for many and this can be seen in the relatively high frequency of two-turn completions for scenario 1. Participants were very successful in the first three scenarios and everyone completed scenario 2 in one turn. It is difficult to say if this is due to its low complexity or because it was their second interaction with the system.

Table 7.3: Frequency of number of turns needed to complete a scenario.

Turns	**Sc. 1**		**Sc. 2**		**Sc. 3**		**Sc. 4**		**Sc. 5**		**All**	
1	10	52.6%	19	100.0%	11	57.9%	7	36.8%	9	47.4%	56	58.9%
2	7	36.8%	0	0.0%	4	21.1%	4	21.1%	0	0.0%	15	15.8%
3	1	5.3%	0	0.0%	2	10.5%	1	5.3%	5	26.3%	9	9.5%
4	1	5.3%	0	0.0%	2	10.5%	1	5.3%	1	5.3%	5	5.3%
5	0	0.0%	0	0.0%	0	0.0%	2	10.5%	3	15.8%	5	5.3%
6	0	0.0%	0	0.0%	0	0.0%	2	10.5%	0	0.0%	2	2.1%
7	0	0.0%	0	0.0%	0	0.0%	1	5.3%	0	0.0%	1	1.1%
8	0	0.0%	0	0.0%	0	0.0%	1	5.3%	0	0.0%	1	1.1%
9	0	0.0%	0	0.0%	0	0.0%	0	0.0%	1	5.3%	1	1.1%
mean	1.632		1.000		1.737		3.105		2.737		2.042	

In the vast majority of cases, scenarios were completed within five turns, and often in much less. Scenarios 4 and 5 are the only scenarios that reach five-turn attempts or more, with outliers of seven, eight, and nine turns needed to complete them. These scenarios are also most complex. Similar strategies are observed among participants in these cases. All three participants that used seven or more attempts to complete scenarios 4 or 5 started using keywords despite their previous experiences, and reordered or adjusted keywords between turns. When this did not improve the results, the researcher reminded them about the system working similarly to human-to-human conversation. Two par-

ticipants then formalized their request by prepending the start of a natural language question, but without changing the keywords already present (e.g., *"please give me invoices Alice 2021"*). This indicates that the prototype did not make its capabilities and expectations crystal clear, because participants seem to think prepending the start of a natural language question would do the job, while this has no effect on the NLU at all. The logged search requests of these three outlying cases are discussed in the following paragraphs.

Participant 6 completed scenario 4 in seven turns and reordered and adjusted keywords before formalizing their request by prepending the start of a natural language question without adjusting the keywords already present. The last two turns were used to formalize the question completely. Interestingly, this participant thought capitalization may have mattered.

1.	"2021 facturen gemeente"	("*2021 invoices municipality*")
2.	"2021 verkoopfacturen gemeente"	("*2021 sales invoices municipality*")
3.	"2021 verkoopfacturen Gemeente"	("*2021 sales invoices Municipality*")
4.	"Verkoopfacturen Gemeente 2021"	("*Sales Invoices Municipality 2021*")
5.	"Doe mij de Verkoopfacturen Gemeente 2021"	("*Give me the Sales Invoices Municipality 2021*")
6.	"Doe mij de Verkoopfacturen van de Gemeente 2021"	("*Give me the Sales Invoices from the Municipality 2021*")
7.	"Doe mij de Verkoopfacturen van de Gemeente van het jaar 2021"	("*Give me the Sales Invoices from the Municipality of the year 2021*")

Participant 11 completed scenario 4 in eight turns and reordered and adjusted keywords before formalizing their request by prepending the start of a natural language question without adjusting the keywords already present.

1.	"facturen gemeente 2021"	("*invoices municipality 2021*")
2.	"gemeente facturen 2021"	("*municipality invoices 2021*")
3.	"alle facturen gemeente"	("*all invoices municipality*")
4.	"all facturen contact gemeente 2021"	("*all invoices contact municipality 2021*")
5.	"gemeente verkoopfacturen 2021"	("*municipality sales invoices 2021*")
6.	"verkoopfacturen gemeente 2021"	("*sales invoices municipality 2021*")
7.	"Ik ben opzoek naar verkoopfacturen gemeente 2021"	("*I am looking for sales invoices municipality 2021*")
8.	"verkoopfacturen van de gemeente van 2021"	("*sales invoices from the municipality of 2021*")

Participant 17 completed scenario 5 in nine turns and reordered and adjusted keywords before asking a question. The ninth turn was needed to include the data granularity.

1.	"consultancy 2021"	("*consultancy 2021*")
2.	"consultancy"	("*consultancy*")
3.	"project consultancy"	("*project consultancy*")
4.	"omzet consultancy"	("*revenue consultancy*")
5.	"omzet consultancy 2021 kwartaal"	("*revenue consultancy 2021 quarter*")
6.	"omzet 2021"	("*revenue 2021*")
7.	"omzet project consultancy 2021"	("*revenue project consultancy 2021*")
8.	"wat was de omzet van het project consultancy in 2021"	("*what was the revenue of project consultancy in 2021*")
9.	"wat was de omzet per kwartaal van het project consultancy in 2021"	("*what was the revenue per quarter of project consultancy in 2021*")

Overall, for scenario 5 fewer participants took a high number of turns than for scenario 4. This suggests a minor level of learning or increased understanding after further use. Scenario 5 shows no completions in two turns. It was observed that many participants that did not complete the scenario in one turn needed multiple turns to address the inclusion of all required information, such as project, time period, and data granularity.

Nineteen participants filled out the post-scenario questionnaire for five scenarios, which resulted in 95 data points for each of 12 variables across five valid values: (1) Strongly Disagree; (2) Disagree; (3) Neither agree nor disagree; (4) Agree; (5) Strongly Agree. All results are found in Appendix J. The median values per variable per scenario are shown in Table J.1. Answers are visualized per variable per scenario in 3D bar charts in Figure J.1 and the median variable value per scenario in 2D bar charts in Figure J.2. Having too few data points makes the quantitative data unreliable for showing if people's perception of searching with natural language significantly changes after five experiences. Besides, the rising level of complexity and non-perfect performance of the prototype's understanding (discussed in Section 7.4) affected participants' perceptions in a way that was not measurable. However, these results (together with observations during the sessions) can explain how people experienced interacting with the prototype overall.

Participants tend to agree with most measures, except future use, after their first encounter with the prototype. While three variables show a slight increase in median value from their first to last ratings, most do not end higher or lower. Results show ratings with high levels of agreement from the start, but this can be explained by the first two scenarios being most simple. The gradual rise in complexity may also explain why the initial rise in ratings for most variables stagnated from the third scenario on. Participants travelling up the learning curve may be offset by the rising complexity. After all scenarios, not a single measure has a lower median agreement than after the initial experience.

Scenarios 3, 4 and 5 were more complex than the first two scenarios. This may explain some of the changes in measures. As shown in Figure 7.1, capability clarity, understand-

ing, and result correctness show a similar pattern where they are more positively perceived after the first scenario, until a decline in agreement after completing scenarios 3 and 4 (where the rise in complexity may have degraded the participants' experience), until they are again more positively perceived after the fifth scenario where participants may have acclimated to the increased complexity. This wave pattern seems to match the changes in mean number of turns participants needed to resolve each scenario (Table 7.3).

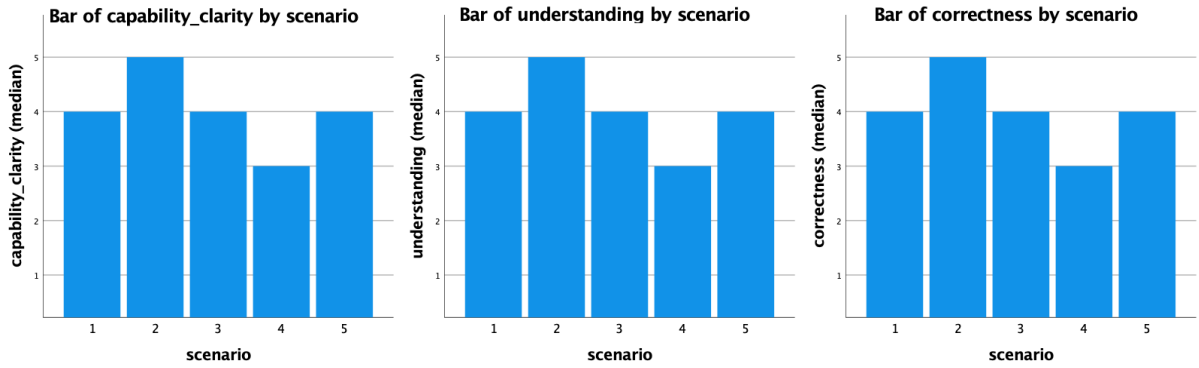

Figure 7.1: 2D bar charts of three measures showing similar patterns in change of perception across scenarios.

Several measures show how participants diverge in ratings across scenarios. In the beginning, during most simple scenarios, they tend to positively agree with these measures, but after more experience with scenarios of increasing complexity, participants have diverged to both more agreement and more disagreement for these measures. In all cases, the counts for 'Strongly Disagree' and 'Strongly Agree' have both ultimately increased. This is shown in the 3D Bar Count charts of expected behaviour, ease of use, and cognitive effort in Figure 7.2. This may show how some of the participants learned the system more than others and/or were able to break away from their conditioning more than others during the interaction while the complexity increased. Who can or cannot keep up may have depended on the application's design and personal factors, as discussed in the results of the thematic analysis. Participants perceived ease of use, cognitive effort, understanding, and response clarity much lower after doing scenario 3. This was also a scenario where some participants tried searching for the textual content of the document in question, which is not supported by the prototype. For ease of use and cognitive effort, overall agreement shifted slightly lower again after doing scenario 4. This was a scenario where many participants stuck to keywords and were not being understood because of it. Some mentioned they thought it understood keyword queries too, because the prototype appeared to understand this format during the first two or three scenarios. However, due to the lower complexity of these scenarios, people entered what can be considered short natural language phrases. However, these measures show recovery after the fourth and/or fifth scenarios, where it seems like many people still managed to have adapted to the prototype's expectations or break away from their conditioning. The interactive feedback was perceived as clear overall and is also indicated by the response clarity measure

leaning towards agreement after five scenarios. Learnability—a participant's perception of how quickly they became familiar with the interaction—peaked after scenario 2. This was the second scenario that also worked well with a keyword-like request. Participants may have thought they knew how the interaction worked, while being put on the wrong foot going into the next scenarios.

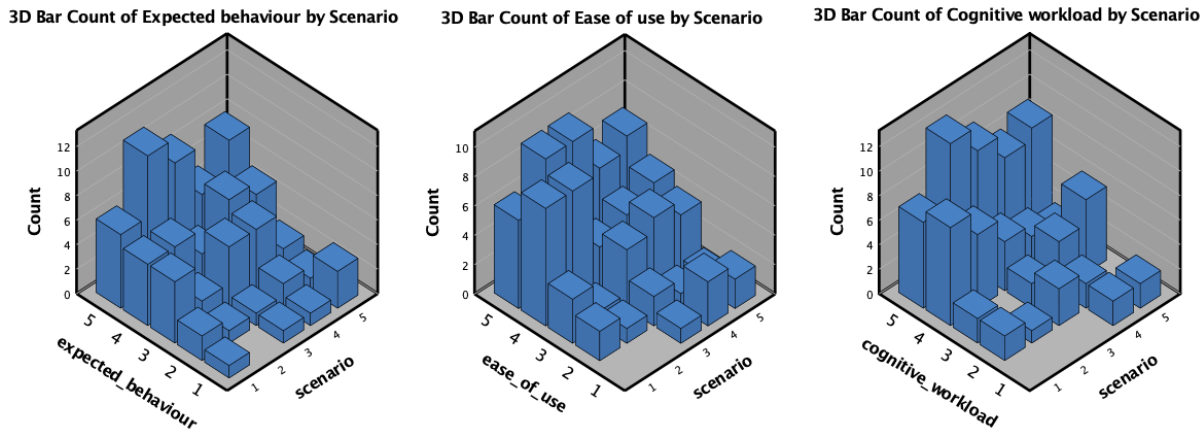

Figure 7.2: 3D Bar Count charts of three measures showing divergence of agreement across scenarios.

7.4 Prototype Evaluation

The results of evaluating the prototype system cover the external models used for POS tagging, dependency tagging, and NER, temporal and money expression tagging, subphrase extraction, ontology mapping, and intent scoring. The number of logged requests produced by the participants during the user study is 284, of which 233 are unique after the prototype's preprocessing step. The two pretrained models used in the prototype system are not evaluated by the author, but metrics reported by the authors of the models are discussed instead. Accuracy evaluation of the pretrained POS and dependency parsing model `nl_core_news_sm`[1] from *spaCy* are reported per task: an accuracy of 0.94 for POS tagging, an F1-score of 0.86 for sentence segmentation, and an accuracy of 0.80 for labelling dependencies. The medium and large models only increase these scores about one percent point each, but do increase the memory requirements. Reported quality of the NER model `ner-dutch-large`[2] from *Flair* that was pretrained on the Dutch *CoNLL-03* dataset are shown in Table 7.4 below. The overall F1-score reported is 0.9258.

Temporal expression tagging and money expression tagging are both implemented similarly using REs. This means evaluation mostly concerns checking if the designed rules covered the language used in the logged requests. Scenarios 2 and 5 require the participant to enter a temporal expression, and only scenario 3 covers a money expression.

[1] `https://spacy.io/models/nl#nl_core_news_sm`
[2] `https://huggingface.co/flair/ner-dutch/blob/main/training.log`

Table 7.4: Reported results of the standard four-class Dutch NER model in *Flair*.

Class	Precision	Recall	F1-score
LOC	0.9580	0.9432	0.9505
MISC	0.9181	0.8694	0.8931
ORG	0.8885	0.9218	0.9048
PER	0.9525	0.9672	0.9598

This means the collected samples to evaluate are much alike and do not cover either algorithm well. For completeness, the accuracy of the temporal expression tagger on the logged requests is 99.6%. These expressions mostly consist of different ways to mention the year 2021, where almost all expressions use the absolute year number. Only one specific expression is not correctly tagged (twice): "*omzet consultancy 2021 kwartaal*". The accuracy of the money expression tagger on the logged requests is 93.1%. The rules for this algorithm omitted three common cases found in the logged requests, causing the expressions not to be tagged: omitting "*euro*" after the number, using a dash to convey a range of amounts (e.g, "*100-150 euro*"), and using an article in front of a number range "*tussen **de** 100 en 150 euro*". Fortunately, the latter case had no effect on slot filling, because subphrase extraction kept the expression intact, despite the tokens not having been merged. In one case, the number digitizer made a mistake where it converted "*honderdvijftig*" into "*100vijftig*" during preprocessing.

Table 7.5 below shows the frequencies of the unique logged requests that were labelled by the author as either more resembling keywords or a natural language request. Making this distinction provides additional insight into the metrics discussed in this section, because the prototype system was designed to handle natural language requests, not keywords. Frequency of keyword search queries is high first, before declining in scenario 3; presumably because the required context needed to get the correct results is difficult to express as keywords alone. The five 'other' requests are unrelated to the scenarios because participants entered those to explore the system.

Table 7.5: Frequency analysis of unique logged requests that either more resemble keywords or a natural language request.

	Sc. 1	Sc. 2	Sc. 3	Sc. 4	Sc. 5	Other	All
Keywords	24	10	12	35	26	0	107
Natural language	16	5	34	34	32	5	126
	60.0%	66.7%	26.1%	50.7%	44.8%	0%	45.9%

Subphrase extraction splits the search request into one or more smaller stretches of tokens using the dependency tree for more focused slot filling. Evaluating the unique logged requests, the dependency trees of the vast majority are small or simple enough that subphrase extraction does not split up the sentence at all (174 requests). Only 29 requests are split into two sub phrases, 19 requests into three subphrases, 7 requests into four subphrases, and 4 requests into five subphrases. Six undesired splits were found in the logged request (97.4% success), five of which come from the 59 requests that were split into two or more subphrases (91.5% success). These consist of cases where adjective and noun were split, verb and subject were split, or tokens were wrongfully ignored. For example, the temporal expression in "*kosten k1 2022*" is lost, and "*toon openstaande en verlopen offertes tussen de 100 en 150 euro*" is split up while both "*openstaande*" and "*verlopen*" are modifying "*offertes*". However, the temporal expression in the first case is incorrectly tagged as punctuation (`punct`) by the dependency parser and therefore ignored in preprocessing, and the latter case would not be split up if the money expression was merged by the tagger (error of the preceding article). Overall there was no clear benefit to subphrase extraction in the user study, because the scenarios did not elicit search requests complex enough to resolve confusion during slot filling (e.g., including multiple contact names or temporal expressions in a single request). More examples of subphrase extraction on logged requests are found in Appendix C.

Ontology mapping is critical to intent determination and slot filling. The breadth of the REs representing each concept determines the flexibility of request understanding. The ontologies and the natural language representations were designed based on the example requests collected among colleagues (Appendix A) and provide a foundation that supports a variety of natural language. However, during the user study some terms were commonly used by participants that were not supported. These terms include "*inkomende facturen*" for purchase invoices ("*inkoopfacturen*"); "*resultaten*", "*kengetallen*" or "*kwartaalcijfers*" for all key figures (of the quarter); and "*crediteuren*" and "*debiteuren*" for sales invoices and purchase invoices, respectively. Using the mapped concepts, intent ranking correctly classified 80.7% (188 : 45) of the unique requests made during the user study. This means it scored the right intent the highest, out of the five defined, but not that all slots were filled correctly. Results per scenario and separated by either more resembling keywords or a natural language request are shown in Table 7.6. The accuracy lies higher when natural language requests were used for almost all scenarios except scenario 2. This is explained by the fact that the prototype is designed to handle natural language requests, not keywords, but reporting on a key figure (scenario 2) triggered well on the keywords too. One non-keyword requests for scenario 2 was not classified correctly because it used an out-of-vocabulary word ("*kengetallen*" instead of e.g., "*kosten*").

Table 7.6: Intent ranking accuracy of all unique logged requests.

	Sc. 1	**Sc. 2**	**Sc. 3**	**Sc. 4**	**Sc. 5**	**Other**	**All**
All	77.5%	93.3%	67.4%	84.1%	84.5%	100.0%	80.7%
Keywords	70.8%	100.0%	33.3%	80.0%	76.9%	n/a	73.8%
Natural language	87.5%	80.0%	79.4%	88.2%	90.6%	100.0%	86.5%

Slot filling extracts the information from the search request that is required to execute the search in the exact way as the user has specified. The SER for the unique requests from the user study are shown in Table 7.7 below (lower is better). The prototype system performed much better on requests that were determined to more resemble a natural language search request than a keyword query. This should not be surprising, because the ontology's natural language representations were designed for these type of requests. The SER for scenario 1 is relatively high because the ontology did not cover the term "*inkomende*" that was commonly used to specifically address purchase invoices. All errors in scenario 3 concern not being able to capture the amount range for the same reasons as discussed in the beginning of this section. A common error found in the natural language requests regarding scenario 4 is using the words "*in*" and "*uit*" to address a date period (e.g., "*facturen van de gemeente uit 2021*"). The system maps this to a `Bookkeeping.Period` concept, but the `Find Invoices` intent requires a more specific type of date to be useful and therefore misses a slot. Task completion rates of all unique logged requests are shown in Table 7.8 below (higher is better). Task completion counts the cases where both intent determination and slot filling were entirely correct.

Table 7.7: Slot Error Rates of all unique logged requests (lower is better).

	Sc. 1	**Sc. 2**	**Sc. 3**	**Sc. 4**	**Sc. 5**	**Other**	**All**
All	0.358	0.133	0.208	0.459	0.248	0.000	0.333
Keywords	0.378	0.200	0.333	0.788	0.484	n/a	0.573
Natural language	0.333	0.000	0.200	0.204	0.088	0.000	0.172

Table 7.8: Task completion rates of all unique logged requests (higher is better).

	Sc. 1	**Sc. 2**	**Sc. 3**	**Sc. 4**	**Sc. 5**	**Other**	**All**
All	0.400	0.800	0.500	0.217	0.414	1.000	0.408
Keywords	0.375	0.800	0.250	0.000	0.077	n/a	0.206
Natural language	0.438	0.800	0.588	0.441	0.688	1.000	0.579

7.5 Improvements to the Interaction Design and Prototype Implementation

The user study also revealed several improvements to the implementation of the interaction design, especially regarding the integration with Moneybird's UI. The following list provides a summary of learnings:

- Selecting an interpretation reveals the results inside the search box, but this limits the user's possibilities of analysing the results. This hinders the user, because selecting an invoice navigates to that page and clears the results when the search box is opened again. Instead, selecting an interpretation should navigate to the dedicated page in Moneybird with the corresponding filters selected. This offers more surrounding features, does not lose state, and saves the cost of developing result visualizations for the search box.
- The interaction design shows one reflected interpretation if the request was understood or simply communicates if it found an incomplete, ambiguous, or unsupported request. Instead, ambiguities may be more proactively resolved by offering multiple automatic completions as alternative interpretations (similar to [29]). This saves the user from manually resolving the ambiguity. This is especially helpful when keywords are entered that likely lack the required context to produce a confident interpretation. For example, after receiving the keyword query "*facturen Sarah*" ("*invoices Sarah*"), the system could show the user two alternative interpretations: "*Facturen verstuurd naar Sarah*" ("*Invoices sent to Sarah*") and "*Facturen ontvangen van Sarah*" ("*Invoices received from Sarah*").
- Participants further commented on the icon and location of the interactive feedback. They mentioned it did not stand out to them and that the icon gave the wrong impression of its functionality. Including a category heading such as *"Do you mean?"* may also improve the impression and help it stand out, especially if an improved design would include multiple alternative interpretations. Additionally, natural language replies did not always use application terminology and did not show when they were loading, causing confusion.
- The current implementation of the combination of existing and prototype search functionalities caused confusion. When the existing search functionality retrieved no results based on the textual content of the search request, it states there were no *other* results found. This led to participants mistakenly thinking that the natural language search was also unsuccessful.

8 Conclusion

The main research question posed in the introduction of this master's thesis was: *"How does a natural language search interaction help or limit users with their information needs in bookkeeping software compared to traditional search?"* Three subquestions concerned a state-of-the-art design of such an interaction, realization of a prototype within Moneybird, and the evaluation of participants' experiences with the prototype interaction:

1. *"What is a state-of-the-art interaction design of a natural language information retrieval system?"*

Based on related work, a state-of-the-art interaction design of a natural language search interaction for personal information retrieval in bookkeeping software was discussed. Chapter 4 detailed its features, fit with the Moneybird application, and how the system would convey its capabilities and expectations to the user.

2. *"How can a prototype of the interaction design using Dutch natural language be realized in the context of Moneybird?"*

A limited prototype of the designed interaction was developed using a frame-based approach that used the legacy method of ontology mapping using REs and intent scoring to achieve intent determination and slot filling. Chapter 5 presented the choices for this approach, the scope of the prototype, and details on its algorithms.

3. *"How can the prototype interaction be evaluated and compared to traditional search?"*

Inspired by related work on evaluating interactions with both natural language search systems and intelligent systems in general, the developed prototype interaction was evaluated according to a scenario-based user study. As discussed in Chapter 6, each scenario was designed to increase in complexity and was presented to the participants as a written backstory. Qualitative observations and comments by the participants were analysed together with quantitative questionnaire data across twelve variables. The prototype system was validated by evaluating its performance on the logged requests from the user study.

The user study conducted in this master's thesis explored the benefits and limitations to the user experience of using natural language in the search process of a bookkeeping

application. The main benefits discussed include: describing items of which the name and location is unknown; alternatively expressing domain-specific language (jargon) or application terminology; expressing specific or 'deeply stored' information by describing complex navigation and filtering; describing information users are unsure exists; and helping users describe their information need more naturally and with less cognitive effort than selecting keywords. The limitations discussed mainly concern conditioning to keyword search and lack of trust in the system's capabilities due to prior experiences and unfamiliarity. Initial expectations are often not aligned and even when they are, conditioning to keywords and unfamiliarity with a new interaction has users interact in manners common to them. This results in uncertainty to the user about the system's expectations and therefore an increase in cognitive effort during use. Other limitations of search using natural language include requests being more intensive to type than keywords for most people (especially inexperienced typists); and expert users being hesitant to adopt new approaches because they think they are probably faster 'the usual way'. A clear introduction may alleviate negative initial expectations, but more importantly, trust can be built with successful use. The system's performance is therefore critical.

Four main themes were identified in Chapter 7: *Trust and initial expectations* describes how the user's trust of the system's expectations and capabilities affects the interaction. Their trust is influenced by their initial expectations, their prior experiences with other systems using natural language, and the performance of the system during use. Lack of trust causes hesitation in request formulation, despite the introduction and hints provided by the system. Users experience increased cognitive effort trying to adhere to a request format they believe is there. Even when initial expectations are aligned, conditioning to keyword-based search and unfamiliarity with a new interaction has users proceed with caution. With sufficient trust, users try both more free-form and more keyword-style requests. *The path of least resistance* describes that choosing to resolve an information need using natural language search is a personal and per-case consideration involving physical effort, cognitive effort, and time spent. These factors are affected by application design, the user's expertise of the application and their typing ability, and the level of detail required to express the information need. The latter depends on the complexity of the information need and the depth of the data in the application. The effort of writing a natural language request can weigh up to the traditional approaches, but this is a consideration that is made again for each individual and information need. *Learning the system and breaking barriers* describes how users express the need to know the interaction and the system's expectations. Participants are conditioned to using keywords. This can be overcome over time through use, if users are correctly understood. However, this requires more than simple capability hints, such as a clear introduction and request suggestions. The latter also being a common search functionality that users have learned to expect. *Naturalness* describes how people feel that using natural language is fitting for an interaction with a speech-enabled device or system resembling a person. When

speaking out loud, information needs are expressed in natural language, but the interaction lacked humanity and people are conditioned to immediately transform this need into keywords. Despite speaking being faster and taking less physical effort than typing, written interactions that feel like talking to a person elicit natural language use.

The quantitative data collected during the study reflected the designed rise in complexity and supported observations made during the user study sessions. The current form of the prototype was found to be difficult to use effectively for many participants. The goal of using natural language in search is to allow users to ask for what they want, rather than having to transform their information needs into intermediary forms such as keywords or paths through the application's navigation. However, the natural language suggestions and interactive feedback were not clear enough, and participants were strongly influenced by their previous experiences with (keyword-based) search systems. Conditioning can be overcome with use, only if the system's NLU performance is good enough. Conversational search can iteratively guide the user towards their answer, but it is then imperative that the system assists the user by offering alternative completions to resolve ambiguities, contradictions, and impossibilities instead of forcing the user to adjust their request to its standards, even when keywords are entered.

8.1 Limitations

Possible alterations to the user study design may be explored. First, scenarios were written on paper and read by the participant. The researcher could read them out loud instead, which may contribute to a more realistic colleague-to-colleague interaction (as portrayed in the scenarios) and more intrinsic natural language use. Second, the sessions were conducted with a demo administration containing dummy data instead of participants' own data to treat the same scenarios. This may have negatively affected the immersion and their natural train of thought. A format can be chosen where participants choose their own information needs to resolve using their own financial administration. Last, some participants that were selected had no prior experience with the application's existing search functionality. Their experience was negatively impacted by limitations of its current implementation. Instead of selecting participants based on their self-selected experience with the entire application, experience with the search functionality should be specifically inquired. Furthermore, a striking number of participants mentioned being handy with technology. This was not intentionally selected nor prevented, and may be a bias of people that were interested in the topic and more willing to participate, or a bias in the population of clients of Moneybird. This is also somewhat reflected in the frequency of use of smart assistants of the remaining nineteen participants, where all invitees that never use these technologies ended up not participating.

Although the scenarios only covered topics that were sufficiently supported by the NLU component of the prototype, some natural language use was not correctly understood during sessions; especially during more complex scenarios. This has impacted participants' experiences. Quantitative data could have provided more insights with more data points. For this purpose, the number was limited because of the combination with an in-depth qualitative analysis. Although divergence in perceptions were seen, going through five scenarios may be insufficient to capture a learning curve as intended, especially when a rise in complexity is included. More scenarios or a longitudinal study may be more appropriate to assess if participants manage to 'break the barrier' of conditioning.

Some features from the interaction design were not implemented in the limited prototype because of technical challenges and time constraints. It is difficult to say how the inclusion of these features would have affected the participants' experiences and thus the results. However, as described in the theme *Learning the system and breaking barriers*, participants mentioned their experience with request suggestions in similar search systems and how it may have positively affected them. Prominent request completion suggestions may more clearly communicate intention and capabilities and lower the physical effort, and so may clear goals and introduction (e.g., translation software has no issues eliciting natural language). During the iterative search process, the ability to critique may compensate for the increased physical effort compared to keywords.

8.2 Future Work and Advice for Continued Development

Chapter 7 discussed that the current implementation of the interaction design was insufficient to convey its capabilities and expectations to the user within a couple of interactions. The relevant parts of the UI and the interaction design of the prototype should be redesigned based on the findings of this thesis. Additionally, described features such as completion suggestions and related search suggestions should be incorporated, because these may aid users significantly based on comments collected during the user study.

The prototype's request understanding performance affected users negatively and should be improved. Improved request understanding performance can be achieved by utilizing an existing LLM and fine-tuning it to the task of intent determination and slot filling for bookkeeping search requests. An LLM such as *GPT-3* is pre-trained on an immense corpus of text, which means it has already learned a lot of information about language and how it is used. This makes it easier to train the model on a specific task, such as intent classification. Especially NLG, such as completion suggestions, related search suggestions, and interactive feedback, can also be achieved using a fine-tuned LLM. An additional benefit of this approach is that this would replace the outside service written in Python and therefore better integrate with the existing Ruby application, which re-

duces development and maintenance efforts. However, the financial costs of fine-tuning and using *GPT-3* via its API should be taken into consideration. In addition, because it is a third-party service, its reliability—now or in the future—cannot be guaranteed. The prototype architecture was purposefully designed to allow for an upgrade of the request understanding component without a major overhaul to the search component in Moneybird. This means continued development could switch to such an approach after sufficient dataset samples have been gathered. All considering, it is the author's recommendation to continue development in this direction for a company of Moneybird's capacity.

Additional adjustments that could be investigated are less related to NLP, but more a subject of IR. First, the system could choose which slots to suggest—to help the user narrow down their initial results—by selecting those with much information gain based on statistics from the database, instead of randomly suggesting unused slots of the selected frame. This may help the user quicker and keep the conversation shorter. Second, the system could use the structured data available in its database to match entities more reliably, instead of using NER during request analysis. This would require additional research on how to feed this structured data to the chosen approach for intent determination and slot filling, such as a fine-tuned LLM.

In closing, it is important to note that implementing a natural language search interaction not only requires an upfront development cost, but also maintaining intent templates and training (or fine-tuning) the neural network again for new or underperforming intents. This may be especially challenging for small and medium enterprises that do not have dedicated data science teams. Although the next step using speech may seem obvious and may solve some of the limitations of typing natural language for search, it brings new challenges such as mishearing, interruptions, pauses, and multiple moves. In addition, the users' behaviours and expectations towards a voice-controlled system may differ from what was found in this thesis.

Bibliography

[1] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 54–59. DOI: 10.18653/v1/N19-4010. [Online]. Available: https://aclanthology.org/N19-4010 (visited on 07/11/2022).

[2] A. Aula and M. Käki, "Understanding Expert Search Strategies for Designing User-Friendly Search Interfaces.," in *Proceedings of the IADIS International Conference WWW/Internet 2003*, Algarve, Portugal, Nov. 2003, pp. 759–762.

[3] M. Bates, "Models of natural language understanding.," *Proceedings of the National Academy of Sciences*, vol. 92, no. 22, pp. 9977–9982, Oct. 1995, Publisher: Proceedings of the National Academy of Sciences. DOI: 10.1073/pnas.92.22.9977. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.92.22.9977 (visited on 04/21/2022).

[4] K. Baxter, C. Courage, and K. Caine, *Understanding your users: a practical guide to user research methods*, Second edition. Amsterdam: Elsevier, Morgan Kaufmann, 2015, OCLC: ocn918928845, ISBN: 978-0-12-800232-2.

[5] j. Brooke, "SUS: A 'Quick and Dirty' Usability Scale," in *Usability Evaluation In Industry*, Num Pages: 6, CRC Press, 1996, ISBN: 978-0-429-15701-1. [Online]. Available: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale.

[6] T. B. Brown *et al.*, *Language Models are Few-Shot Learners*, arXiv:2005.14165 [cs], Jul. 2020. DOI: 10.48550/arXiv.2005.14165. [Online]. Available: http://arxiv.org/abs/2005.14165 (visited on 12/05/2022).

[7] J. Caulfield, *How to Do Thematic Analysis | Step-by-Step Guide & Examples*, NL, Sep. 2019. [Online]. Available: https://www.scribbr.com/methodology/thematic-analysis/ (visited on 10/18/2022).

[8] A. X. Chang and C. Manning, "SUTime: A library for recognizing and normalizing time expressions," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 3735–3740. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2012/pdf/284_Paper.pdf (visited on 05/06/2022).

[9] Cybersecurity Ventures, "The 2020 Data Attack Surface Report," Cybersecurity Ventures, New York, NY, USA, Tech. Rep., Jun. 2020, p. 5. [Online]. Available: https://info.arcserve.com/hubfs/The%202020%20Data%20Attack%20Surface%20Report/The%202020%20Data%20Attack%20Surface%20Report%20%7C%20Arcserve.pdf (visited on 06/28/2022).

[10] B. Desmet and V. Hoste, "Fine-grained Dutch named entity recognition," *Language Resources and Evaluation*, vol. 48, no. 2, pp. 307–343, Jun. 2014, ISSN: 1574-020X, 1574-0218. DOI: 10.1007/s10579-013-9255-y. [Online]. Available: http://link.springer.com/10.1007/s10579-013-9255-y (visited on 04/21/2022).

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423 (visited on 04/20/2022).

[12] L. Dong and M. Lapata, "Language to Logical Form with Neural Attention," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 33–43. DOI: 10.18653/v1/P16-1004. [Online]. Available: https://aclanthology.org/P16-1004 (visited on 04/29/2022).

[13] M. Dubiel, M. Halvey, L. Azzopardi, and S. Daronnat, "Investigating How Conversational Search Agents Affect User's Behaviour, Performance and Search Experience," in *Proceedings of ACM SIGIR CAIR Workshop*, New York, NY, USA, 2018, p. 8.

[14] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins, "Stuff I've seen: A system for personal information retrieval and re-use," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ser. SIGIR '03, New York, NY, USA: Association for Computing Machinery, Jul. 2003, pp. 72–79, ISBN: 978-1-58113-646-3. DOI: 10.1145/860435.860451. [Online]. Available: http://doi.org/10.1145/860435.860451 (visited on 06/28/2022).

[15] F. Fahmi, K. Tanjung, F. Nainggolan, B. Siregar, N. Mubarakah, and M. Zarlis, "Comparison study of user experience between virtual reality controllers, leap motion controllers, and senso glove for anatomy learning systems in a virtual reality environment," *IOP Conference Series: Materials Science and Engineering*, vol. 851, no. 1, p. 012 024, May 2020, Publisher: IOP Publishing, ISSN: 1757-899X. DOI: 10.1088/1757-899X/851/1/012024. [Online]. Available: https://doi.org/10.1088/1757-899x/851/1/012024 (visited on 10/13/2022).

[16] A. Figueroa, "Exploring effective features for recognizing the user intent behind web queries," *Computers in Industry*, vol. 68, pp. 162–169, Apr. 2015, ISSN: 0166-3615. DOI: 10.1016/j.compind.2015.01.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361515000159 (visited on 09/02/2022).

[17] M. Fowler, *Patterns of enterprise application architecture*, ser. The Addison-Wesley signature series. Boston: Addison-Wesley, 2003, ISBN: 978-0-321-12742-6.

[18] B. Ghai, Q. V. Liao, Y. Zhang, R. Bellamy, and K. Mueller, "Explainable Active Learning (XAL): Toward AI Explanations as Interfaces for Machine Teachers," in *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, Dec. 2020, pp. 1–28. DOI: 10.1145/3432934. [Online]. Available: https://dl.acm.org/doi/10.1145/3432934 (visited on 05/11/2022).

[19] K. Guinee, M. B. Eagleton, and T. E. Hall, "Adolescents' Internet Search Strategies: Drawing upon Familiar Cognitive Paradigms When Accessing Electronic Information Sources," *Journal of Educational Computing Research*, vol. 29, no. 3, pp. 363–374, Oct. 2003, Publisher: SAGE Publications Inc, ISSN: 0735-6331. DOI: 10.2190/HD0A-N15L-RTFH-2DU8. [Online]. Available: https://doi.org/10.2190/HD0A-N15L-RTFH-2DU8 (visited on 05/17/2022).

[20] W. Guo *et al.*, "Deep Natural Language Processing for Search and Recommender Systems," en, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA: ACM, Jul. 2019, pp. 3199–3200, ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3332290. [Online]. Available: https://dl.acm.org/doi/10.1145/3292500.3332290 (visited on 07/22/2022).

[21] D. Hakkani-Tür *et al.*, "Multi-Domain Joint Semantic Frame Parsing Using Bi-Directional RNN-LSTM," in *Interspeech 2016*, ISCA, Sep. 2016, pp. 715–719. DOI: 10.21437/Interspeech.2016-402. [Online]. Available: https://www.isca-speech.org/archive/interspeech_2016/hakkanitur16_interspeech.html (visited on 05/03/2022).

[22] M. Honnibal and I. Montani, "SpaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017.

[23] How-To Geek, *How to Use Natural Language Search in OS X's Spotlight*, Jun. 2016. [Online]. Available: https://www.howtogeek.com/258608/how-to-use-natural-language-search-in-os-xs-spotlight/ (visited on 06/28/2022).

[24] S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer, "Learning a Neural Semantic Parser from User Feedback," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 963–973. DOI: 10.18653/v1/P17-1089. [Online]. Available: https://aclanthology.org/P17-1089 (visited on 04/26/2022).

[25] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. draft. Jan. 2022. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/ (visited on 04/14/2022).

[26] Y. Kammerer and M. Bohnacker, "Children's web search with Google: The effectiveness of natural language queries," in *Proceedings of the 11th International Conference on Interaction Design and Children*, ser. IDC '12, New York, NY, USA: Association for Computing Machinery, Jun. 2012, pp. 184–187, ISBN: 978-1-4503-1007-9. DOI: 10.1145/2307096.2307121. [Online]. Available: http://doi.org/10.1145/2307096.2307121 (visited on 05/16/2022).

[27] E. Kaufmann and A. Bernstein, "Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases," *Journal of Web Semantics*, Semantic Web Challenge 2009, vol. 8, no. 4, pp. 377–393, Nov. 2010, ISSN: 1570-8268. DOI: 10.1016/j.websem.2010.06.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570826810000582 (visited on 05/17/2022).

[28] J.-K. Kim, G. Tur, A. Celikyilmaz, B. Cao, and Y.-Y. Wang, "Intent detection using semantically enriched word embeddings," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, Dec. 2016, pp. 414–419. DOI: 10.1109/SLT.2016.7846297.

[29] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," *Proceedings of the VLDB Endowment*, vol. 8, no. 1, pp. 73–84, Sep. 2014, ISSN: 2150-8097. DOI: 10.14778/2735461.2735468. [Online]. Available: http://doi.org/10.14778/2735461.2735468 (visited on 05/17/2022).

[30] J. Lim and K.-H. Lee, "Constructing composite web services from natural language requests," *Journal of Web Semantics*, vol. 8, no. 1, pp. 1–13, Mar. 2010, ISSN: 1570-8268. DOI: 10.1016/j.websem.2009.09.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570826809000493 (visited on 04/14/2022).

[31] B. Liu and I. Lane, "Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling," *arXiv:1609.01454 [cs]*, Sep. 2016, arXiv: 1609.01454. [Online]. Available: http://arxiv.org/abs/1609.01454 (visited on 05/02/2022).

[32] D. Lukovnikov, A. Fischer, and J. Lehmann, "Pretrained Transformers for Simple Question Answering over Knowledge Graphs," in *The Semantic Web – ISWC 2019*, vol. 11778, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2019, pp. 470–486, ISBN: 978-3-030-30792-9. DOI: 10.1007/978-3-030-30793-6_27. [Online]. Available: http://link.springer.com/10.1007/978-3-030-30793-6_27 (visited on 04/26/2022).

[33] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. New York: Cambridge University Press, 2008, OCLC: ocn190786122, ISBN: 978-0-521-86571-5.

[34] R. Modi, K. Naik, T. Vyas, S. Desai, and S. Degadwala, "E-mail autocomplete function using RNN Encoder-decoder sequence-to-sequence model," in *2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Dec. 2021, pp. 710–714. DOI: 10.1109/ICECA52323.2021.9675961.

[35] S. Mohammed, P. Shi, and J. Lin, "Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 291–296. DOI: 10.18653/v1/N18-2047. [Online]. Available: https://aclanthology.org/N18-2047 (visited on 04/26/2022).

[36] P. Morville and J. Callender, *Search patterns: Design for Discovery*, English. 2010, OCLC: 1250324253, ISBN: 978-1-306-80922-1.

[37] N. Okazaki and J. Tsujii, "Simple and Efficient Algorithm for Approximate Dictionary Matching," in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 851–859. [Online]. Available: https://aclanthology.org/C10-1096 (visited on 04/14/2022).

[38] V. Plachouras *et al.*, "Interacting with Financial Data using Natural Language," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, ser. SIGIR '16, New York, NY, USA: Association for Computing Machinery, Jul. 2016, pp. 1121–1124, ISBN: 978-1-4503-4069-4. DOI: 10.1145/2911451.2911457. [Online]. Available: http://doi.org/10.1145/2911451.2911457 (visited on 04/13/2022).

[39] F. Radlinski and N. Craswell, "A Theoretical Framework for Conversational Search," in *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, ser. CHIIR '17, New York, NY, USA: Association for Computing Machinery, 2017, pp. 117–126, ISBN: 978-1-4503-4677-1. DOI: 10.1145/3020165.3020183. [Online]. Available: http://doi.org/10.1145/3020165.3020183 (visited on 12/19/2022).

[40] S. Ravuri and A. Stoicke, "A comparative study of neural network models for lexical intent classification," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec. 2015, pp. 368–374. DOI: 10.1109/ASRU.2015.7404818.

[41] C. Rosset *et al.*, "Leading Conversational Search by Suggesting Useful Questions," in *Proceedings of The Web Conference 2020*, ser. WWW '20, New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 1160–1170, ISBN: 978-1-4503-7023-3. DOI: 10.1145/3366423.3380193. [Online]. Available: http://doi.org/10.1145/3366423.3380193 (visited on 12/19/2022).

[42] T. Russell-Rose and T. Tate, Eds., *Designing the search experience: the information architecture of discovery*. Amsterdam: Elsevier, Morgan Kaufmann, 2013, ISBN: 978-0-12-396981-1.

[43] I. Ruthven, "Interactive Information Retrieval," *Annual Review of Information Science and Technology*, vol. 42, pp. 43–92, 2008, ISSN: 0066-4200. [Online]. Available: https://strathprints.strath.ac.uk/34250/4/strath_cis_publication_2230.pdf (visited on 06/28/2022).

[44] S. Schweter and A. Akbik, "FLERT: Document-Level Features for Named Entity Recognition," arXiv, Tech. Rep. arXiv:2011.06993, May 2021, arXiv:2011.06993 [cs] type: article. DOI: 10.48550/arXiv.2011.06993. [Online]. Available: http://arxiv.org/abs/2011.06993 (visited on 07/11/2022).

[45] J. Smienk, "Natural Language Search in Accountancy Software," Research Topics, University of Twente, Enschede, The Netherlands, Jul. 2022.

[46] Stanford, *Stanford CoreNLP*, original-date: 2013-06-27T21:13:49Z, Jan. 2022. [Online]. Available: https://github.com/stanfordnlp/CoreNLP (visited on 05/06/2022).

[47] J. Strötgen and M. Gertz, "Multilingual and cross-domain temporal tagging," *Language Resources and Evaluation*, vol. 47, no. 2, pp. 269–298, Jun. 2013, ISSN: 1574-020X, 1574-0218. DOI: 10.1007/s10579-012-9179-y. [Online]. Available: http://link.springer.com/10.1007/s10579-012-9179-y (visited on 05/06/2022).

[48] S. Stumpf, E. Sullivan, E. Fitzhenry, I. Oberst, W.-K. Wong, and M. Burnett, "Integrating rich user feedback into intelligent user interfaces," in *Proceedings of the 13th international conference on Intelligent user interfaces - IUI '08*, Gran Canaria, Spain: ACM Press, 2008, p. 50, ISBN: 978-1-59593-987-6. DOI: 10.1145/

1378773.1378781. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1378773.1378781 (visited on 05/10/2022).

[49] J. R. Trippas, D. Spina, L. Cavedon, H. Joho, and M. Sanderson, "Informing the Design of Spoken Conversational Search: Perspective Paper," in *Proceedings of the 2018 Conference on Human Information Interaction&Retrieval - CHIIR '18*, New Brunswick, NJ, USA: ACM Press, 2018, pp. 32–41, ISBN: 978-1-4503-4925-3. DOI: 10.1145/3176349.3176387. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3176349.3176387 (visited on 06/29/2022).

[50] F. Ture and O. Jojic, "No Need to Pay Attention: Simple Recurrent Neural Networks Work!" In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2866–2872. DOI: 10.18653/v1/D17-1307. [Online]. Available: https://aclanthology.org/D17-1307 (visited on 04/26/2022).

[51] R. Vīksna and I. Skadiņa, "Multilingual Transformers for Named Entity Recognition," en, *Baltic Journal of Modern Computing*, vol. 10, no. 3, 2022, ISSN: 22558950. DOI: 10.22364/bjmc.2022.10.3.18. [Online]. Available: http://www.bjmc.lu.lv/fileadmin/user_upload/lu_portal/projekti/bjmc/Contents/10_3_18_Viksna.pdf (visited on 12/21/2022).

[52] M. Walker, C. Kamm, and D. Litman, "Towards developing general models of usability with PARADISE," *Natural Language Engineering*, vol. 6, no. 3-4, pp. 363–377, Sep. 2000, Publisher: Cambridge University Press, ISSN: 1469-8110, 1351-3249. DOI: 10.1017/S1351324900002503. [Online]. Available: http://www.cambridge.org/core/journals/natural-language-engineering/article/towards-developing-general-models-of-usability-with-paradise/EB8F05325362DBFEAA755931CC6 (visited on 05/18/2022).

[53] Q. Wang, C. Nass, and J. Hu, "Natural Language Query vs. Keyword Search: Effects of Task Complexity on Search Performance, Participant Perceptions, and Preferences," in *Human-Computer Interaction - INTERACT 2005*, M. F. Costabile and F. Paternò, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2005, pp. 106–116, ISBN: 978-3-540-31722-7. DOI: 10.1007/11555261_12.

[54] J. Weizenbaum, "ELIZA—a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/365153.365168. [Online]. Available: https://dl.acm.org/doi/10.1145/365153.365168 (visited on 04/22/2022).

[55] S. Whittaker and C. Sidner, "Email overload: Exploring personal information management of email," in *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96*, Vancouver, British Columbia,

Canada: ACM Press, 1996, pp. 276–283, ISBN: 978-0-89791-777-3. DOI: 10.1145/238386.238530. [Online]. Available: http://portal.acm.org/citation.cfm?doid=238386.238530 (visited on 10/28/2022).

[56] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, "Scalable Zero-shot Entity Linking with Dense Entity Retrieval," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 6397–6407. DOI: 10.18653/v1/2020.emnlp-main.519. [Online]. Available: https://aclanthology.org/2020.emnlp-main.519 (visited on 04/26/2022).

[57] W. Yang *et al.*, "End-to-End Open-Domain Question Answering with BERTserini," *Proceedings of the 2019 Conference of the North*, pp. 72–77, 2019, arXiv: 1902.01718. DOI: 10.18653/v1/N19-4013. [Online]. Available: http://arxiv.org/abs/1902.01718 (visited on 05/11/2022).

[58] K. Yu, S. Berkovsky, R. Taib, J. Zhou, and F. Chen, "Do I trust my machine teammate? an investigation from perception to decision," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ser. IUI '19, New York, NY, USA: Association for Computing Machinery, Mar. 2019, pp. 460–468, ISBN: 978-1-4503-6272-6. DOI: 10.1145/3301275.3302277. [Online]. Available: http://doi.org/10.1145/3301275.3302277 (visited on 05/18/2022).

[59] X. Yu *et al.*, "Dataset and Enhanced Model for Eligibility Criteria-to-SQL Semantic Parsing," in *Proceedings of the 12th Conference on Language Resources and Evaluation*, Marseille, France: European Language Resources Association, May 2020, pp. 5829–5837. [Online]. Available: https://par.nsf.gov/servlets/purl/10168884 (visited on 04/26/2022).

[60] X. Zhang and H. Wang, "A joint model of intent determination and slot filling for spoken language understanding," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16, New York, New York, USA: AAAI Press, Jul. 2016, pp. 2993–2999, ISBN: 978-1-57735-770-4. (visited on 05/02/2022).

[61] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft, "Towards Conversational Search and Recommendation: System Ask, User Respond," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18, New York, NY, USA: Association for Computing Machinery, 2018, pp. 177–186, ISBN: 978-1-4503-6014-2. DOI: 10.1145/3269206.3271776. [Online]. Available: https://doi.org/10.1145/3269206.3271776 (visited on 12/19/2022).

A Internal Exploratory Survey Results

A survey was distributed among colleagues at Moneybird to ask them how they would ask a described natural language search interaction about certain topics or information needs. This led to the following list of 180 responses, grouped by 10 mentioned entities of information need, open coded by the author. The entities that were mentioned in the sample requests include: key figures, invoices, contacts, projects, products, hours, transactions, ledger accounts, estimates, and informational questions. Twenty-three responses were ambiguous and could not be used.

Key figures (43)

"omzet in q1"

"omzet kwartaal"

"wat was mijn omzet vorig jaar"

"welke periode van het jaar heeft de hoogste omzet"

"welke periode van het jaar de minste omzet"

"In welk jaar had ik de meeste omzet?"

"Heb ik dit kwartaal meer omzet dan vorig kwartaal?"

"Omzet van vorige maand"

"Wat is mijn gemiddelde omzet per maand/week in het afgelopen jaar"

"In welke maand had ik de minste omzet"

"Wat is mijn omzet?"

"Geboekte omzet in januari"

"Ik wil graag het btw-rapport zien van het afgelopen kwartaal"

"overzicht van de omzet van de afgelopen drie jaar"

"overzicht van mijn belastingaangiftes van dit jaar"

"Laat mij mijn balans zien"

"open balans 2020"

"Hoeveel winst heb ik gemaakt in kwartaal 1?"

"Maak ik winst?"

"Hoeveel zou ik moeten verdienen om winstgevend te zijn"

"winst afgelopen periode"

"omzet verwachting"

"belasting betalen in q1"

"hoeveel btw betalen"

"krijg ik geld terug van de belasting"

"wat moet ik afdragen aan de Belastingdienst"

"hoeveel btw moet ik komende maand betalen"

"hoeveel btw betaal ik gemiddeld per maand/jaar"

"hoeveel belasting betaald ik gemiddeld per maand"

"Hoe veel btw moet ik afdragen?"

"Hoeveel btw moet ik afdragen?"

"Hoeveel btw moet ik betalen"

"Hoeveel belasting heb ik betaald in 2020"

"Hoeveel btw heb ik betaald voor de btw-aangifte van Q1 2022?"

"Hoeveel btw heb ik in openstaande correcties?"

"Hoeveel btw mag ik terugvragen?"

"Positieve cashflow?"

"Cashflow"

"heb ik dit kwartaal voldoende omzet"

"kwartaal met beste omzet"

"beste omzet kwartaal"

"meer omzet dan vorig jaar"

Invoices (41)

"zoek factuur X" (62)

"Wat is mijn debiteurenstand?"

"Alle inkoopfacturen binnengekomen via Peppol"

"Verlopen facturen tussen januari en maart van dit jaar onder de 300 euro"

"facturen van klant X uit januari"

"Welke facturen moet ik uiterlijk deze week betalen?"

"openstaande facturen mei 2022"

"geef me alle betaalde facturen"

"ik wil alle betaalde facturen inzien"

"welke facturen hebben nog de status openstaand"

"voor welke facturen moet ik een herinnering versturen"

"factuur klanten België"

"facturen van contactpersoon X"

"Alle facturen van 'Contact"'

"Alle verlopen facturen van Contact X"

"Heeft Simon geld overgemaakt?"

"Facturen van T-Mobile die ik nog moet betalen."

"Alle facturen van T-Mobile"

"alle facturen die zijn verzonden door gebruiker X"

"Alle facturen groter dan 100 euro"

"Alle openstaande facturen van 100 euro of meer"

"Alle verlopen facturen van 100 euro of meer"

"alle inkoopfacturen van 500 euro of meer"

"alle verkoopfacturen van 500 euro of meer"

"Verkoopfacturen met de workflow 'Standaard"'

"factuur telefoon"

"facturen met product X" (90)

"facturen met woord X"

"Facturen met minimaal 1 regel met categorie 'ongecategoriseerde inkomsten"'

"Facturen van januari"

"facturen van 2022"

"creditfacturen van 2022"

"Alle facturen uit Q4"

"Alle facturen van vorige jaar"

"Facturen van volgend jaar"

"Facturen van aankomend jaar"

"Alle openstaande facturen van dit jaar"

"Alle facturen die verzonden zijn op 3 april"

"facturen project X"

"Welke facturen hebben nog geen project?"

"Hoeveel facturen maak ik per maand?"

Contacts (32)

"Zoek Praxis"

"contacten verkoop product X" (104)

"contacten met facturen boven de € 1000 in de maand maart"

"welke klant was het meest winstgevend in 2022"

"welke klant was het minst winstgevend in 2022"

"iedereen bij KPN"

"contacten bedrijf X"

"contactpersoon bedrijf X"

"Contacten met verlopen facturen"

"Contacten met meer dan 2 herinnerde, dubieuze of oninbare facturen"

"Contacten die nog openstaande facturen hebben"

"Alle contacten die mij een factuur hebben gestuurd die ik nog moet betalen"

"contacten met een afname van 500 euro of meer"

"totale opbrengst contact X"

"Welke contact heeft X facturen"

"Alle contacten die mij een factuur hebben gestuurd die ik nog moet betalen"

"terugkerende klanten"

"wanneer voor het laatst klant X gefactureerd"

"klanten die product X gekocht hebben"

"Wat is mijn duurste leverancier?"

"Wat zijn mijn top 5 klanten?"

"Wat is mijn grootste klant?"

"contacten in Duitsland"

"Contacten uit Enschede"

"Contacten in regio Amsterdam"

"alle klanten met postcode XXXX (en in 10 km omtrek)"

"Nederlandse klanten"

"alle debiteuren"

"alle crediteuren"

"contacten met betaalmethode 'SEPA"'

"contacten zonder mandaat"

"contacten waarvan het btw-nummer ongeldig is"

Projects (4)

"Projecten met de meeste omzet deze maand"

"Hoeveel omzet heb ik voor project X in kwartaal 1 2022"

"project dat het meeste geld opbracht"

"welk project is het duurst"

"welk project meeste klanten"

"projecten met omzet boven de 1000 euro"

"alle projecten met omzet onder de 1000 euro"

"welke projecten waren winstgevend"

"hoeveel projecten hadden we dit jaar"

"Wat zijn mijn meest winstgevende projecten?"

"Welk project verdient het minst?"

"Projecten met de minste omzet"

"welk project of product heeft de hoogste marge"

"in welk project heb ik de hoogste marge gedraaid"

"welk project had dit jaar de hoogste omzet"

"Wat is mijn omzet voor project X"

"Wat zijn mijn best presterende projecten?"

"Hoeveel uren heeft persoon X geboekt per projecten"

"welk project bijna aan max. aantal uren"

"goedlopende projecten"

"hoe was de verdeling van de projecten over het jaar"

"projecten die al gefactureerd zijn"

"alle projecten in 2021"

"projecten gestart in Q3-2021"

"projecten met woord X"

"Resultatenrekening van project X"

"laat de resultatenrekening zien van project X"

"welke projecten duurden het langst"

"Aan welke projecten heb ik minder dan 40 uur gewerkt?"

"Projecten met uren die ik nog moet factureren"

"Welke projecten hebben nog geen facturen?"

"Projecten waar minimaal 1 factuur gebruik van maakt"

Products (5)

"welk project of product heeft de hoogste marge"

"welk product heeft de hoogste marge"

"welk product heb ik het meest verkocht"

"welk product heeft de meeste omzet gedraaid"

"Welke producten gaven de meeste omzet in kwartaal 1?"

Hours (4)

"Hoe veel niet-declarabele uren heb ik dit kwartaal gewerkt?"

"Hoeveel uren heb ik voor klant X gewerkt?"

"Hoeveel declarabele uren heb ik deze maand gewerkt?"

"Hoeveel niet-declarabele uren heb ik vorige week geschreven"

Transactions (2)

"banktransacties die nog niet gekoppeld zijn"

"Hoeveel transacties moet ik nog zelf koppelen?"

"Een overzicht van onverwerkte banktransacties."

Ledger Accounts (1)

"Op welke categorie is de meeste omzet geboekt?"

Estimates (1)

"welke offertes heb ik in 2021 verzonden"

Informational Questions (22)

"Hoe maak ik een credit-nota?"

"Wat moet ik nog invullen voordat ik btw aangifte kan doen"

"moet ik icp-aangifte doen"

"wanneer moet ik aangifte doen"

"wat zijn de btw maanden"

"Hoe voeg ik een btw tarief toe?"

"Wat moet ik nog doen om mijn btw aangifte compleet te maken?"

"Hoe kan ik mijn pakket verhogen"

"welke correcties zijn er voor de btw geweest in afgelopen kwartaal"

"Hoe zeg ik mijn abonnement op?"

"Hoe kan ik mijn account verwijderen"

"Hoe voeg ik een product of abonnement toe"

"Waarom staat er een bedrag op een rubriek onder 3b?"

"Wanneer moet ik weer aangifte doen?"

"Help"

"help me"

"wat doe ik fout"

"wat is een leidraad qua omzet/kosten in de eerste jaren bij de start van een onderneming"

"wie kan mij helpen met de inrichting van mijn boekhouding"

"wat is belangrijk als ik start met ondernemen"

"wat zijn de grootste valkuilen"

"Wat betekend dit X btw-bedrag?"

B POS and dependency labels used by *spaCy*

Table B.1: List of POS and their meaning.

POS tag	Meaning	Example
adj	adjective	last, next, open, least, average, profitable
apd	adposition	in, of, under, above, per
adv	adverb	why, what, how
aux	auxiliary verb	was, are, is
cconj	coordinating conjunction	and
det	determiner	the, this, that, which, all
intj	interjection	hmm, aah, boo
noun	noun	invoice, today, revenue, quarter, euros
num	numeral	2021, three
part	particle	call up
pron	pronoun	I, my
propn	proper noun	January
punct	punctuation	., ?, :, /
sconj	subordinating conjunction	than
sym	symbol	Q1, X, €
verb	verb	have, want
x	other	–

Table B.2: List of dependency labels and their meaning.

Dependency label	Meaning
acl	clausal modifier of a noun (adnominal clause)
acl:relcl	relative clause modifier
advcl	adverbial clause modifier
advmod	adverbial modifier
advmod:emph	emphasizing word, intensifier

Continued on next page

Table B.2 (continued)

Dependency label	**Meaning**
advmod:lmod	locative adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
aux:pass	passive auxiliary
case	case-marking
cc	coordinating conjunction
cc:preconj	preconjunct
ccomp	clausal complement
clf	classifier
compound	compound
compound:lvc	light verb construction
compound:prt	phrasal verb particle
compound:redup	reduplicated compounds
compound:svc	serial verb compounds
conj	conjunct
cop	copula
csubj	clausal subject
csubj:pass	clausal passive subject
dep	unspecified dependency
det	determiner
det:numgov	pronominal quantifier governing the case of the noun
det:nummod	pronominal quantifier agreeing in case with the noun
det:poss	possessive determiner
discourse	discourse element
dislocated	dislocated elements
expl	expletive
expl:impers	impersonal expletive
expl:pass	reflexive pronoun used in reflexive passive
expl:pv	reflexive clitic with an inherently reflexive verb
fixed	fixed multiword expression
flat	flat multiword expression
flat:foreign	foreign words
flat:name	names
goeswith	goes with
iobj	indirect object

Continued on next page

Table B.2 (continued)

Dependency label	**Meaning**
list	list
mark	marker
nmod	nominal modifier
nmod:poss	possessive nominal modifier
nmod:tmod	temporal modifier
nsubj	nominal subject
nsubj:pass	passive nominal subject
nummod	numeric modifier
nummod:gov	numeric modifier governing the case of the noun
obj	object
obl	oblique nominal
obl:agent	agent modifier
obl:arg	oblique argument
obl:lmod	locative modifier
obl:tmod	temporal modifier
orphan	orphan
parataxis	parataxis
punct	punctuation
reparandum	overridden disfluency
root	root
vocative	vocative
xcomp	open clausal complement

C Subphrase Extraction Examples

Subphrase extraction divides a search request into one or more smaller segments using the dependency tree. The goal is to isolate related information. For example, splitting up multiple temporal expressions and their subjects. Here, examples are given of subphrase extraction on example search requests. Each example includes the dependency tree and the resulting subphrases.

Example 1

Dutch: "*openstaande en verlopen offertes die vorige week verstuurd zijn tussen de 100 en 150 euro*"

English: "*open and due estimates that were sent last week between 100 en 150 euros*"

Subphrase extraction using the tree visualized in Figure C.1 results in the following desired subphrases:

- "*openstaande en verlopen offertes*" ("*open and due estimates*")
- "*die vorige week verstuurd*" ("*that sent last week*")
- "*tussen 100 en 150 euro*" ("*between 100 en 150 euros*")

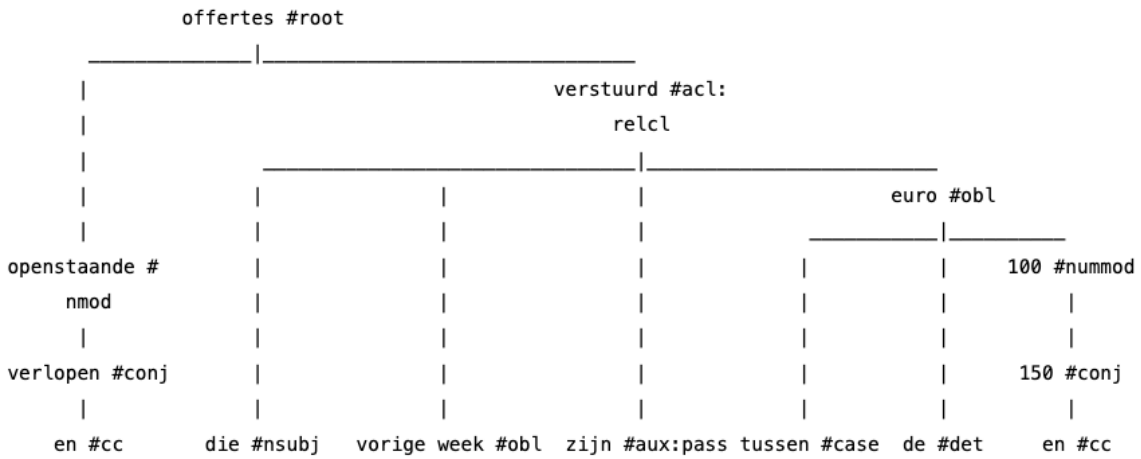

Figure C.1: Dependency tree of example request 1.

Example 2

Dutch: "*welke facturen verlopen aankomende maand, waren afgelopen kwartaal verstuurd en zijn deze maand betaald?*"

English: "*what invoices are due next month, were sent last quarter and are paid this month?*"

Subphrase extraction using the tree visualized in Figure C.2 results in the following desired subphrases:

- "*welke facturen verlopen aankomende maand*" ("*what invoices are due next month*")
- "*afgelopen kwartaal verstuurd*" ("*sent last quarter*")
- "*en deze maand betaald*" ("*and paid this month*")

In this case, three temporal expressions and their corresponding types of dates are isolated in separate subphrases.

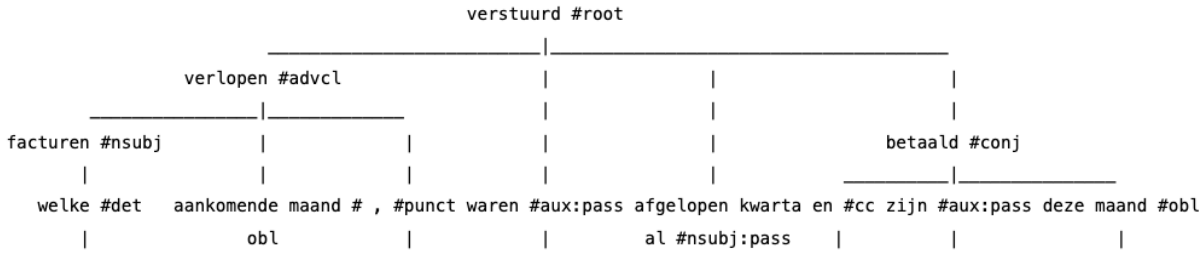

Figure C.2: Dependency tree of example request 2.

Example 3 (undesired)

Dutch: "*facturen uit 2021 verstuurd aan de gemeente*"

English: "*invoices from 2021 sent to the municipality*"

Subphrase extraction using the tree visualized in Figure C.3 results in the following **undesired** subphrases:

- "*facturen uit 2021 verstuurd*" ("*invoices from 2021 sent*")
- "*aan gemeente*" ("*to municipality*")

In this case, the verb was separated from its subject. Desired outcomes would be not dividing the short and simple sentence, or having "*verstuurd*" be included in the subphrase with the contact.

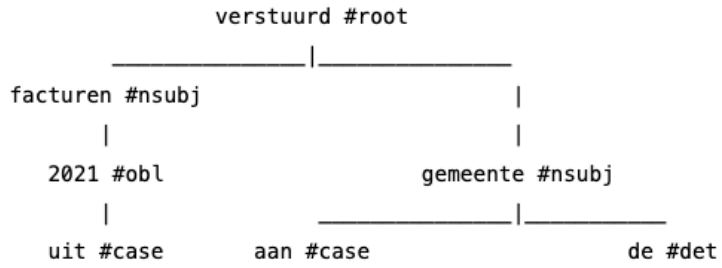

Figure C.3: Dependency tree of example request 3.

Example 4 (undesired)

Dutch: "*welke facturen zijn 4 jaar geleden verstuurd en betaald op 25 mei 2015?*"

English: "*which invoices were sent 4 years ago and paid on 25 May 2015*"

Subphrase extraction using the tree visualized in Figure C.4 results in the following **undesired** subphrases:

- "*welke facturen 4 jaar geleden verstuurd en betaald op 25 mei 2015*" ("*which invoices sent 4 years ago and paid on 25 May 2015*")

Different temporal expressions and their types of dates are not separated because of an inaccurate parse tree. Looking at the dependency tree, the verb "*betaald*" should point to the date "*op 25 mei 2015*". Explicitly adding a comma to the search request produces a more accuracy dependency tree, as shown in Figure C.5, and the following desired subphrases:

Dutch: "*welke facturen zijn 4 jaar geleden verstuurd, en betaald op 25 mei 2015?*"

English: "*which invoices were sent 4 years ago, and paid on 25 May 2015*"

- "*welke facturen verstuurd*" ("*which invoices sent*")
- "*4 jaar geleden verstuurd*" ("*sent 4 years ago*")
- "*en betaald op 25 mei 2015*" ("*and paid on 25 May 2015*")

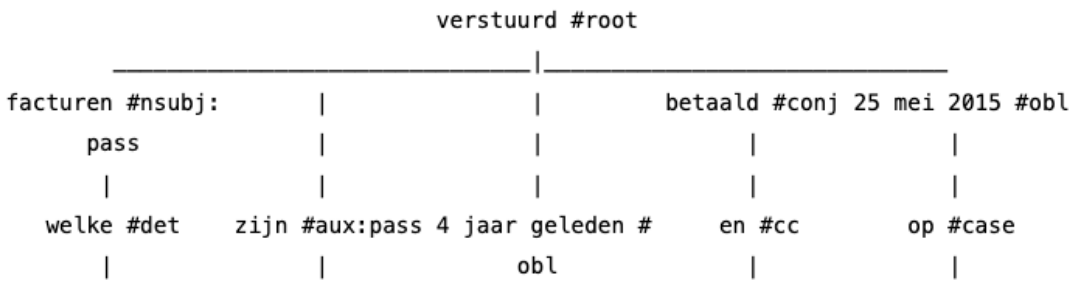

Figure C.4: Dependency tree of example request 4a.

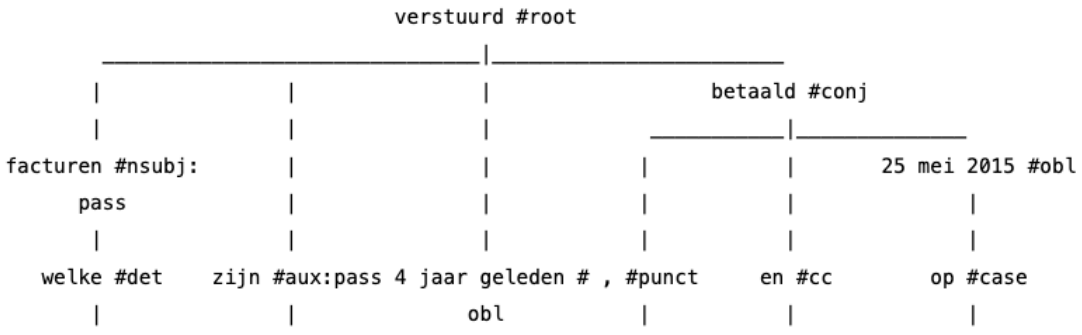

Figure C.5: Dependency tree of example request 4b.

Example 5

Dutch: "*inkomende facturen die openstaan of verlopen zijn*"

English: "*incoming invoices that are open or due*"

Subphrase extraction using the tree visualized in Figure C.6 results in the following desired subphrases:

- "*inkomende facturen*" ("*incoming invoices*")
- "*die openstaan of verlopen*" ("*that are open or due*")

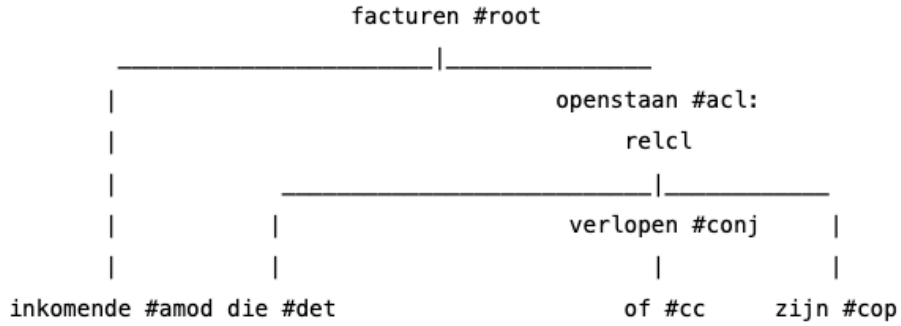

Figure C.6: Dependency tree of example request 5.

Example 6

Dutch: "*geef me de omzet voor project consultancy voor vorig jaar per kwartaal*"

English: "*give me the revenue for project consultancy for last year per quarter*"

Subphrase extraction using the tree visualized in Figure C.7 results in the following desired subphrases:

- "*geef me*" ("*give me*")
- "*geef per kwartaal*" ("*give per quarter*")
- "*omzet*" ("*revenue*")

- "*omzet voor vorig jaar*" ("*revenue for last year*")
- "*voor project consultancy*" ("*for project consultancy*")

This case results in a high number of subphrases, but this is not undesired if isolated information belongs to different slots.

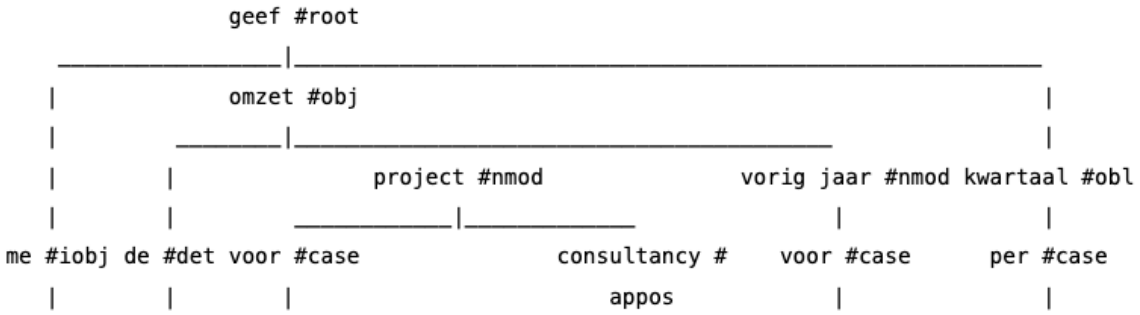

Figure C.7: Dependency tree of example request 6.

Example 7

Dutch: "*wat was de omzet per kwartaal van het project consultancy in 2021?*"

English: "*what was the revenue per quarter of project consultancy in 2021?*"

Subphrase extraction using the tree visualized in Figure C.8 results in the following desired subphrases:

- "*wat*" ("*what*")
- "*omzet in 2021*" ("*revenue in 2021*")
- "*per kwartaal*" ("*per quarter*")
- "*van project consultancy*" ("*of project consultancy*")

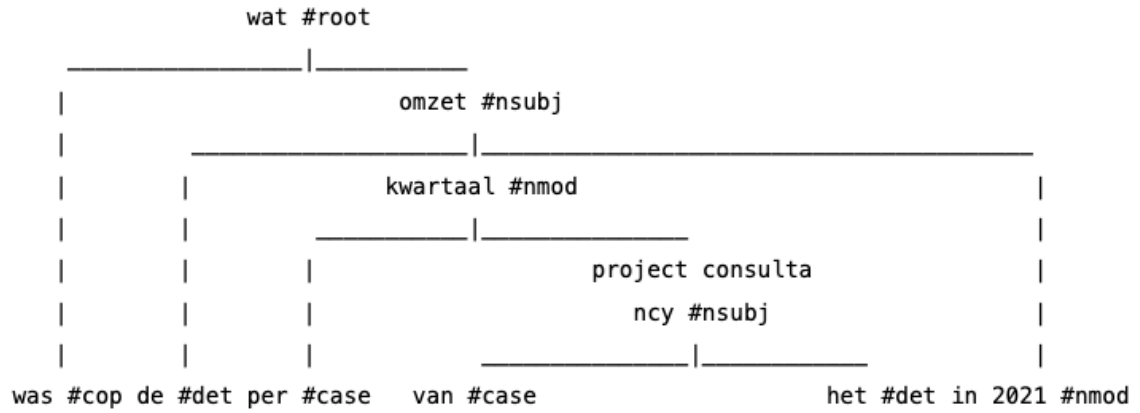

Figure C.8: Dependency tree of example request 7.

Example 8

Dutch: "*verlopen facturen van vorige week die verliepen op vierentwintig augustus 2022*"

English: "*overdue invoices from last week that were due on twenty-four August 2022*"

Subphrase extraction using the tree visualized in Figure C.9 results in the following desired subphrases:

- "*verlopen facturen*" ("*overdue invoices*")
- "*facturen van vorige week*" ("*invoices from last week*")
- "*die verliepen op 24 augustus 2022*" ("*that were due on twenty-four August 2022*")

Again in this case, two temporal expressions are separated in different subphrases to isolate the type of date with the corresponding temporal expression.

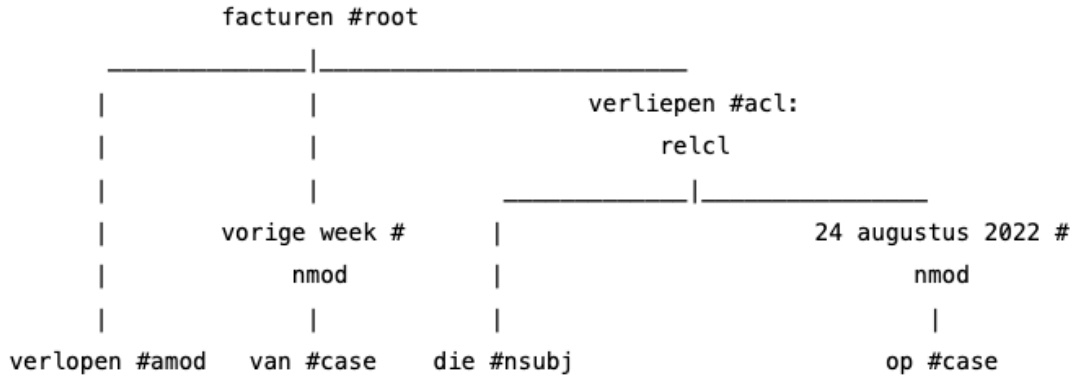

Figure C.9: Dependency tree of example request 8.

D Prototype Example of Defined Template for Reply Generation

The following code listing shows a segment of YAML defined string localization used in the application to define the language used to formulate replies for the `Find Invoices` intent.

Listing D.1: Segment of YAML defined string localization used in the prototype to construct natural language replies for a single intent.

```
find_invoices_scope:
  ambiguity_scope:
    due_before_date: "Er zijn geen %{invoice_type} die verlopen voordat ze toegevoegd zijn."
    due_before_sent: "Er zijn geen %{invoice_type} die verlopen voordat ze verzonden zijn."
    invalid_state: "Sorry, %{invoice_type} %{invalid_state}."
    invalid_state_scope:
      remindable: "kun je geen herinnering voor sturen"
      reminded: "kun je geen herinnering voor hebben verstuurd"
      uncollectible: "kunnen niet oninbaar zijn"
      unsend: "kunnen niet onverzonden zijn"
    paid_before_date: "Er zijn geen %{invoice_type} die betaald zijn voordat ze toegevoegd zijn."
    paid_before_sent: "Er zijn geen %{invoice_type} die betaald zijn voordat ze verzonden zijn."
    paid_in_future: "Er zijn geen %{invoice_type} die in de toekomst betaald zijn."
    sales_and_purchase: "Bedoel je alleen verkoopfacturen, inkoopfacturen of beide?"
    sent_before_date: "Er zijn geen %{invoice_type} die verzonden zijn voordat ze toegevoegd zijn."
    sent_in_future: "Er zijn geen %{invoice_type} die in de toekomst verstuurd zijn."
    state_future_date: "%{state} %{invoice_type} kunnen geen factuurdatum in de toekomst hebben."
  reply_no_results: "ik heb helaas geen%{state} %{invoice_type} kunnen vinden%{appendix}."
```

```
19    reply_results: "ik heb %{count}%{state} %{invoice_type} gevonden%{
          appendix}."
20    reply_understanding: "%{state} %{invoice_type}%{appendix}."
21    slot_scope:
22      amount_scope:
23        base: "met een bedrag %{amount}"
24        between: "tussen %{lower} en %{upper}"
25        eq: "van exact %{money}"
26        ge: "van %{money} of meer"
27        gt: "van meer dan %{money}"
28        le: "van %{money} of minder"
29        lt: "van minder dan %{money}"
30      contact: "die '%{v}' als contact hebben"
31      date_due: "die %{v} verlopen"
32      date_finalized: "die %{v} toegevoegd zijn"
33      date_paid: "die %{v} betaald zijn"
34      date_sent: "die %{v} verstuurd zijn"
35      invoice_type_scope:
36        both: "verkoop- en inkoopfacturen"
37        purchase: "inkoopfacturen"
38        sales: "verkoopfacturen"
39      state_scope:
40        late: "verlopen"
41        paid: "betaalde"
42        remindable: "waarvan je een herinnering kan sturen"
43        reminded: "waarvan een herinnering verstuurd is"
44        sent: "verzonden"
45        uncollectible: "oninbare"
46        unpaid: "onbetaalde"
47        unsend: "onverzonden"
```

E Copy of User Study Recruitment Form (Dutch)

Title:

Zoeken in 'gewone taal'! Deelnemen aan een onderzoek bij Moneybird op kantoor.

Description:

Beste Moneybird gebruiker,

Mijn naam is Jeroen en voor mijn afstuderen onderzoek ik een slimme zoekfunctie in Moneybird. Dit betekent dat je zinnen of vragen zou kunnen typen in 'gewone taal'. Alsof je tegen een persoon praat!

Om dit te testen zoek ik gebruikers die mee willen doen aan een gebruikersonderzoek bij Moneybird op kantoor in Enschede. We doorlopen een aantal scenario's om met het prototype te zoeken wat je wilt vinden in Moneybird. Ik ben enorm benieuwd naar jouw ervaringen!

Ben je geïnteresseerd? Vul dan dit formulier in. De antwoorden gebruiken we om een diverse selectie aan deelnemers te maken. Alvast ontzettend bedankt!

Jeroen

- Als dank voor je deelname krijg je een Coolblue-cadeaubon t.w.v. € 50,- en reiskostenvergoeding.
- Het onderzoek duurt maximaal een uur.
- Midden september ga ik een diverse selectie maken van de aanmeldingen. Als je bent geselecteerd krijg je een uitnodiging om een afspraak te plannen voor het onderzoek. Het onderzoek vindt plaats in begin oktober op ons kantoor in Enschede aan Moutlaan 35.

Questions:

Hoe vaak gebruik je 'slimme' assistenten zoals Google Home, Siri, Alexa, of klantcontact chatbots?

- ❏ Nooit
- ❏ Bijna nooit
- ❏ Wel een keer per maand
- ❏ Wel een keer per week
- ❏ Dagelijks

Hoe lang gebruik je al Moneybird?

- ❏ minder dan 6 maanden
- ❏ 6 maanden - 1 jaar
- ❏ 1 jaar - 5 jaar
- ❏ 5 jaar of langer

Ik gebruik Moneybird om...

- ❏ Facturen te versturen.
- ❏ Offertes te versturen.
- ❏ Uren te boeken.
- ❏ Inkomende documenten te verwerken.
- ❏ Mijn banktransacties te koppelen.
- ❏ Projecten bij te houden.
- ❏ Mijn resultaten in te zien (bv. omzet/verlies/winst).
- ❏ BTW-aangifte in te doen.

Hoe dichtbij Enschede woon je?

- ❏ dichterbij dan 25 km
- ❏ 26 - 65 km
- ❏ 66 - 100 km
- ❏ 100 km of verder weg

Wat is je leeftijdsgroep?

- ❏ jonger dan 18 jaar
- ❏ 18 - 29 jaar
- ❏ 30 - 39 jaar
- ❏ 40 - 49 jaar
- ❏ 50 - 59 jaar
- ❏ 60 jaar of ouder

Wat is je naam?

Op welk e-mailadres kan ik je bereiken?

F User Study Consent Form (Dutch)

starts on the next page.

Toestemmingsformulier

Gebruikersonderzoek naar zoeken in Moneybird met gewone taal.

Contactgegevens onderzoeker
naam: Jeroen Smienk
e-mail: j.smienk@student.utwente.nl

Contactgegevens supervisor
naam: dr. Shenghui Wang
e-mail shenghui.wang@utwente.nl

Contactgegevens onafhankelijk ethiek comité UT
e-mail: ethicscommittee-cis@utwente.nl

Leest u deze informatie alstublieft goed door:
Dit onderzoek wordt uitgevoerd door Jeroen Smienk, master student aan Universiteit Twente, als onderdeel van het afstuderen bij Moneybird. Het doel van dit onderzoek is om de voor- en nadelen te onderzoeken van het zoeken in een financiële administratie met "gewone taal". U zult gebruik maken van een prototype dat is toegevoegd aan Moneybird om verschillende zoektaken uit te voeren. Zoals het vinden van bepaalde facturen, transacties of contacten. De zoektaken hebben als doel verschillende delen van het prototype te onderzoeken. Hierna vult u een vragenlijst in over uw ervaringen met systeem. Zoals gebruiksvriendelijkheid, duidelijkheid, snelheid en werking. U kunt de taken niet halen of falen en uw invoer is nooit fout. Het onderzoek duurt maximaal 1 uur.

Over uw deelname:

- Uw deelname brengt geen fysieke, financiële of juridische risico's met zich mee.
- Deelname is vrijwillig en u mag op elk moment besluiten te stoppen en u hoeft niets te doen waar u niet mee instemt; of vragen te beantwoorden die u niet wilt beantwoorden.
- U krijgt een financiële tegemoetkoming voor uw deelname.

In dit onderzoek wordt u: **geobserveerd**; **geïnterviewd**; en gevraagd een vragenlijst in te vullen. Dit gebeurt tijdens: een **één-op-één gesprek** en **experiment** waarbij u op een computer zoektaken uitvoert in Moneybird op het kantoor van Moneybird.

De volgende gegevens worden verzameld:

- **audio** dat wordt opgenomen tijdens het gesprek en het experiment (wat u zegt kan worden omgezet in tekst; zoals anonieme quotes);
- **observaties** die worden genoteerd door de onderzoeker (bv. als een scenario lastig blijkt of als u een interessante manier bedenkt om de taak uit te voeren);
- **persoonsgegevens** en **antwoorden** die u invult onderaan dit toestemmingsformulier en in de vragenlijst over uw ervaring met het systeem tijdens het onderzoek. Bijvoorbeeld uw leeftijdsgroep en voorkennis.

De resultaten van het onderzoek worden uitsluitend anoniem gebruikt voor het afstuderen van de onderzoeker en ter verbetering van Moneybird. Uitsluitend geanonimiseerde onderzoeksresultaten worden gepubliceerd; dus geen audio-opnames (deze worden direct na analyse verwijderd). Alle resultaten inclusief uw persoonsgegevens op dit toestemmingsformulier worden opgeslagen in een vertrouwelijk gedeelte van de beveiligde cloud die Moneybird gebruikt. Persoonsgegevens worden verwijderd na het afstuderen van de onderzoeker. U heeft het recht om al eerder Moneybird te verzoeken uw persoonsgegevens in te laten zien, te laten wijzigen of al eerder te laten verwijderen. U kunt dit doen door contact op te nemen met support@moneybird.nl.

Vinkt u alstublieft de volgende hokjes af en tekent u daaronder:

Ik heb de informatie over dit gebruikersonderzoek gelezen of het is mij voorgelezen en ik heb dit begrepen. Ik heb de mogelijkheid gehad vragen te stellen en deze zijn naar tevredenheid beantwoord.	☐
Ik begrijp dat ik niets hoef te doen dat ik niet wil en geen vragen hoeft te beantwoorden die ik niet wil beantwoorden.	☐
Ik begrijp dat ik op elk moment kan stoppen met deelnemen aan dit onderzoek zonder opgave van reden.	☐
Ik begrijp dat deelname aan dit gebruikersonderzoek betekent dat ik binnen het kantoor van Moneybird geobserveerd wordt, geïnterviewd wordt en dat ik achter een computer zoektaken ga uitvoeren terwijl de audio in de ruimte opgenomen word.	☐
Ik begrijp dat onderzoeksresultaten ook gebruikt kunnen worden om bestaande elementen van Moneybird te verbeteren of toekomstige functionaliteit te bepalen.	☐
Ik begrijp dat verzamelde gegevens inclusief persoonsgegevens worden opslagen in een vertrouwelijk gedeelte van de beveiligde cloud die Moneybird gebruikt en dat persoonsgegevens na het afstuderen van de onderzoeker worden verwijderd.	☐
Ik begrijp dat ik contact op kan nemen met Moneybird support om al eerder persoonsgegevens over mijn deelname in te zien, te laten wijzigen of te laten verwijderen.	☐
Ik neem vrijwillig deel aan dit gebruikersonderzoek.	☐

Handtekening	**Naam**	**Datum**
_____	_____	_____

G Copy of User Study Instructor Protocol

1. Introduction. (∼5 min)
   (a) Sign consent form, if not done so yet. (∼2 min)
   (b) Explain how we will proceed. (∼2 min)
   (Prototype will not cover everything or work perfectly. Not the point.)
   (c) Expectations of them.(∼1 min)
   (No right or wrong answers. The goal is to find out what helps or limits them with this way of working.)
   Start audio
2. Ask about initial expectations. (∼3 min)
3. Go through scenarios. (∼45 min)
   (a) Let participant read backstory. (∼1 min)
   (b) Traditional approach. (∼2.5 min)
   (c) Prototype approach. (∼2.5 min)
   Note turns
   (d) Let participant fill in post-scenario questionnaire. (∼3 min)
4. Post-test discussion and collect demographics. (∼7 min)
   (a) Ask how their experiences relate to their initial expectations. (∼3 min)
   (What helped or limited them?)
   (b) Ask if they can think of future use cases. (∼3 min)
   (c) Prior knowledge and demographics. (∼1 min)
   Stop audio
5. Debrief, handing out incentives, and goodbye. (∼5 min)

H Copy of User Study Participant Handout (Dutch)

Scenario 1 (presented on separate printouts)

Een nieuwe werkdag is aangebroken en je zit klaar om je administratie bij te werken. Je vindt het belangrijk om je inkomende facturen op tijd te betalen en zoekt daarom deze die je bedrijf nog niet betaald heeft.

Scenario 2

Om je voor te bereiden op een telefonisch gesprek morgen, zoek je een van de kengetallen uit je administratie van het eerste kwartaal van 2022. Je bent benieuwd wat je kosten waren. Zoek dit bedrag op.

Scenario 3

Je collega Marja kwam bij je om wat te vragen over een offerte van een tijdje geleden. Ze wil graag contact opnemen met de klant omdat ze nooit gereageerd hadden. Misschien dat ze alsnog behoefte hebben aan een schilderklus. Ze wist alleen niet meer uit haar hoofd om welke offerte het precies ging. Marja dacht dat de prijs tussen honderd en honderdvijftig euro lag. Zoek deze offerte op zodat Marja dit contact opnieuw kan benaderen.

Scenario 4

Je bent gevraagd een interne presentatie te geven over werkzaamheden die jullie vorig jaar verricht hebben voor de gemeente. Een onderdeel van dit verslag beslaat de facturen die hen gestuurd zijn. Zoek deze facturen op zodat je ze kan verwerken in de presentatie.

Scenario 5

Vorig jaar heb je advies gegeven aan verscheidene klanten in de vorm van interactieve online workshops. Je overweegt om deze workshops weer te geven voor een aantal klanten dit komende jaar. Je hebt de resultaten destijds geboekt onder een project met de naam ‘consultancy’. Je wilt weten hoeveel omzet je toen gemiddeld per fiscaal kwartaal geboekt hebt. Zoek dit bedrag op.

I Copy of User Study Post-scenario Questionnaire (Dutch)

Participant ID

_____

Scenario #

_____

Het systeem werkt zoals ik verwacht. (expected behaviour)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik kan gemakkelijk vinden wat ik wilde zoeken. (ease of use)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik hoef me niet veel in te spannen om het systeem te gebruiken. (cognitive workload)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Het is mij telkens duidelijk wat ik het systeem kan vragen. (capability clarity)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik vind dat het systeem mij goed begrijpt. (understanding)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik vind de reactie van het systeem gemakkelijk te begrijpen. (response clarity)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik vind dat het systeem mij juiste resultaten geeft op mijn vragen. (correctness)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik vind het tempo van de interactie prettig. (pace)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik vind dat het systeem snel reageert op mijn verzoeken. (speed)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik heb snel door hoe het systeem werkt en hoe ik het gebruik. (learnability)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik ben tevreden over de manier van werken met dit systeem. (satisfaction)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

Ik wil het systeem vaker gebruiken. (future use)

Helemaal oneens, oneens, niet oneens/eens, eens, Helemaal eens

J Quantitative Results of the User Study Post-Scenario Questionnaire

Nineteen participants filled out the post-scenario questionnaire for five scenarios, resulting in 95 data points for each of 12 variables across five values: (1) Strongly Disagree; (2) Disagree; (3) Neither agree nor disagree; (4) Agree; (5) Strongly Agree.

Table J.1: Median value of each variable per scenario.

Measure	Sc. 1	Sc. 2	Sc. 3	Sc. 4	Sc. 5
Expected behaviour	4	5	4	4	4
Ease of use	4	4	4	4	4
Cognitive effort	4	5	5	4	4
Capability clarity	4	5	4	3	4
Understanding	4	5	4	3	4
Response clarity	4	5	4	4	5
Result correctness	4	5	4	3	4
Interaction pace	5	5	5	5	5
Response speed	5	5	5	5	5
Learnability	4	5	4	4	4
Satisfaction	4	5	5	5	5
Future use	3	4	4	4	4

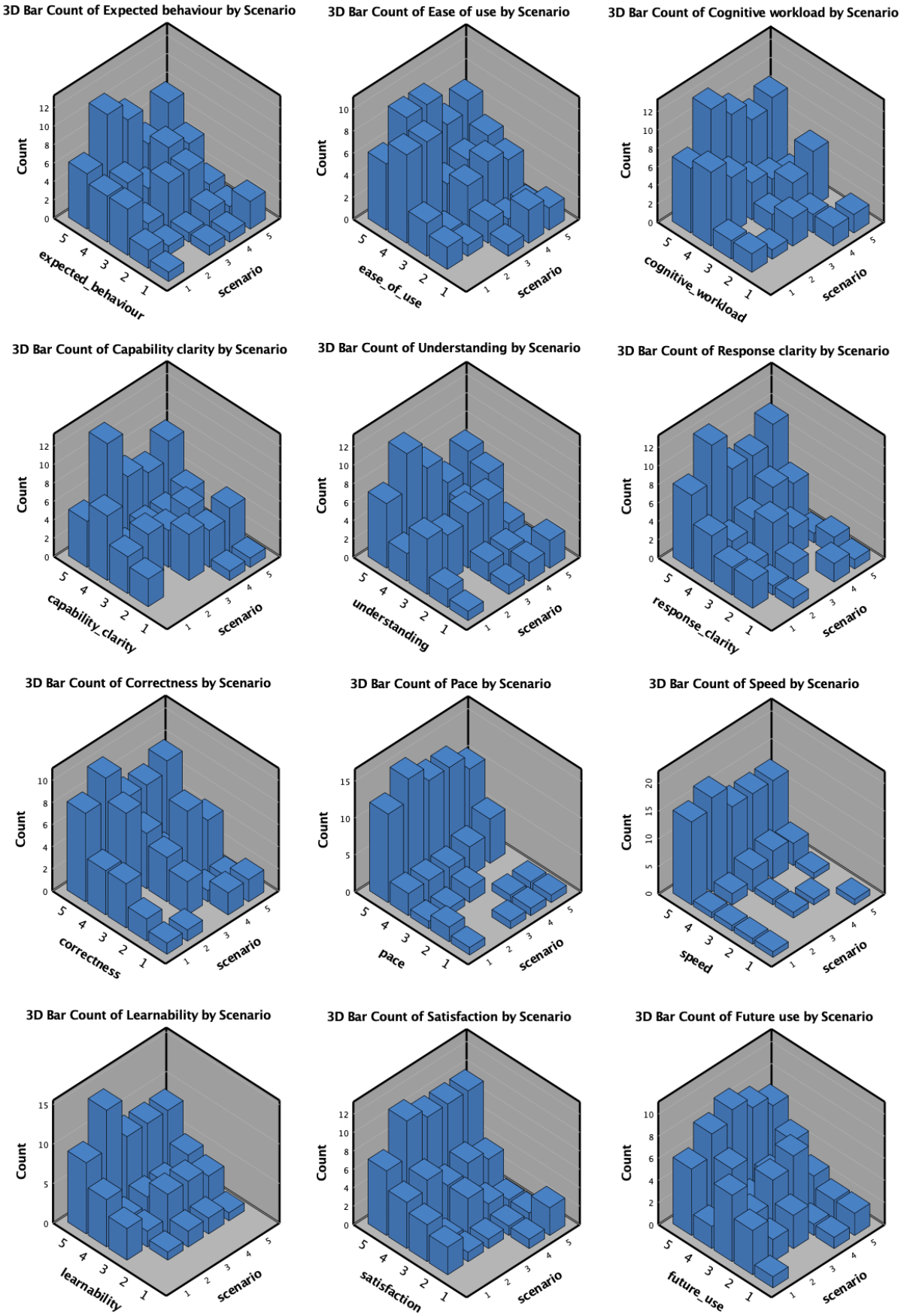

Figure J.1: 3D Bar Count graphs for each quantitative variable per scenario (higher variable values mean more positive perceptions).

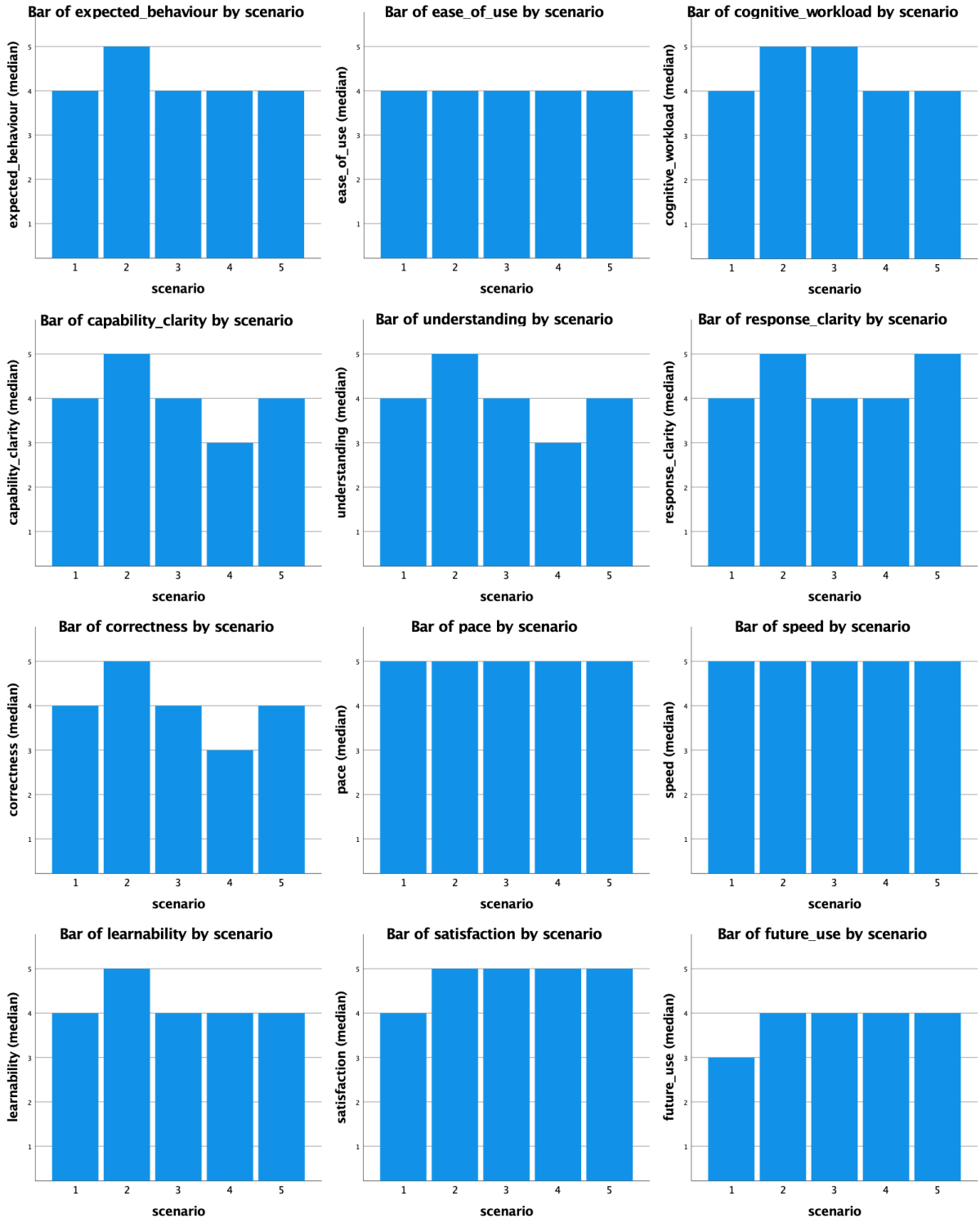

Figure J.2: 2D Bar graphs for median of each quantitative variable per scenario (higher variable values mean more positive perceptions).