Politecnico di Torino

Politecnico di Torino

Data Science and Engineering

A.y. 2023/2024

Degree session 04/2024

Large Language Models for Ambiguity Detection and Resolution in Smart-Homes

Advisors:

Luigi De Russis

Tommaso Calò

Candidate:

Iván Darío Contreras Pérez

Abstract

This thesis studies the use of LLMs within smart home systems for ambiguity detection and resolution. Existing literature often overlooks the significance of user-centric disambiguation, as smart home systems typically interpret user intentions based on context and predefined settings without considering the user's specific definitions of ambiguous concepts (e.g., "large," "cozy," "comfortable"). This oversight potentially impacts the accuracy of results from the user's perspective.

This thesis explores how users perceive the interaction with a text-based smart-home system and its results when user-oriented disambiguation is set in place. To achieve this, an add-on system was developed to identify concept ambiguities in user intents and present disambiguation options in both text and image formats using LLMs and multiple prompting strategies.

Experiments involving 7 participants were conducted within a simulated smart-home environment, with four predefined intents. The results indicate that users view disambiguation positively, as it reduces the probability space of responses and consequently increases the accuracy. However, it was found that the introduction of the disambiguation add-on decreases the precision. Despite this, users perceive the trade-off favorably, valuing increased accuracy even at the expense of precision; responses aligned closely with the user's context and mental flow were rated higher, even when they deviated from the user's initial intent expectation.

Acknowledgements

Detrás de esta tesis se esconde más que mi esfuerzo académico; hay un tejido de experiencias, tanto positivas como negativas, que he atravesado durante este tiempo, las cuales, afortunadamente, me han conducido a la culminación de este proyecto. Cada una de ellas ha sido compartida con personas maravillosas que he encontrado en mi camino, y tambien con aquellas que siempre han estado incluso a la distancia.

Quiero agradecer a mi esposa por ser mi apoyo más cercano, a pesar de estar separados por un océano, su respaldo incondicional se refleja en cada palabra escrita aquí. A mis padres y hermanas, por enseñarme a seguir lo que me hace feliz y nunca hacerme dudar de mis capacidades. A Isor, por ser un amigo constante y estar presente tanto en los momentos de alegría como en las adversidades. Y a Edgar y Pablo, por ser esos amigos que siempre estan, y que brindan apoyo en la medida de lo posible.

Tambien a todas las personas que aunque por períodos cortos, han aportado cosas positivas en mi camino hacia la culminación de la maestría, Simone (B8), Nicole, Remi, Fatemeh, Abdollah, y a los que no nombre pero tengo presentes, gracias por regalarme momentos de alegria.

Finalmente, Luigi y Tommaso, les doy las gracias por su paciencia y seguimiento de la tesis a distancia.

Table of Contents

List of Figures

Glossary

SHS
: Smart Home System

LLM
: Large Language Model

HCI
: Human Computer Interaction

NLP
: Natural Language Processing

NER
: Named Entity Recognition

RNN
: Recurrent Neural Network

LSTM
: Long-Short Term Memory Neural Network

BRNN
: Bidirectional Recurrent Neural Network

CoT
: Chain of Thought

IFTTT
: If This Then That

Chapter 1

Introduction

In recent years, the integration of Natural Language Processing within smart home systems has marked a significant advancement in Human-Computer Interaction (HCI) paradigms. These systems, leveraging sophisticated algorithms and techniques, aim to interpret user intents accurately to provide seamless and intuitive interactions; examples of them are integrated in our daily basis, from phones and laptops to fridges and space shuttles. However, an essential aspect often overlooked is the user's own definition of ambiguous concepts within their interactions with these systems.

For instance, a user giving an intent to set the speaker volume to a "quiet" level might expect around 10% of volume if he lives in a quiet neighborhood, on the other hand if the neighborhood became crowded and noisy 30% of volume might be expected; with this in mind, what exactly does "quiet" mean for the system? The conventional approach in smart home systems involves interpreting user intentions only based on context and predefined settings, downplaying the role of individual interpretations that users assign to ambiguous concepts such as "large", "quiet", or "comfortable". This standardized interpretation, while efficient in many cases, might fail to capture the real preferences and expectations of individual users, leading to discrepancies between the system's outputs and the user's desired outcomes.

Multiple solutions for this problem have raised, however most of them are focused on the system and its capability to identify relevant aspects of the environment and the user; improving the ability of the system to translate the ambiguous intent into actions on the devices, to understand the semantics of the intents to provide a more accurate response or even to reason about a complex intent and create a detailed plan to achieve a goal. The interest in the user interaction with the system and its own vision of the world throughout the time have been ignored in these solutions. The importance of changing this resides in the fact that the user satisfaction is the final goal of any smart home system, a satisfaction that highly depends on the accuracy of the response and the user perception of the system.

Consequently, the aim of this thesis is to know how users perceive the introduction of a disambiguation strategy that eliminates this standard interpretation of the concepts taking into account the way users consume content and the integration that can be achieved with existing smart home systems. Additionally, this thesis explores the capabilities of LLMs to manage tasks that require a high level of semantic interpretation like disambiguation and their integration with third party services and tools.

1.1 Aim and Limits

This thesis has an exploratory nature, it does not aim to establish a baseline for the usage of LLMs inside smart home systems, or checking the optimal approach to solve the stated problem. The core goal of the thesis is focused on the user experience and the multiple ways LLMs can be used to solve the recurrent problem of ambiguities in the interaction with the smart home systems. To accomplish this multiple objectives are necessary since the disambiguation strategy should be implemented and tested by the users to gather the necessary data to reach a useful conclusion from the thesis.

1.2 Objectives

- Establish a disambiguation strategy that integrates well with the existing smart home systems, has a clear purpose, is easy to understand and use, is simple and is usable by a diverse user base.
- Implement the disambiguation strategy taking advantage of the LLMs capabilities through multiple prompting strategies and third party tools integration.
- Evaluate the user perception using surveys and experiments with the disambiguation system as research instruments.

1.3 Thesis Overview

The thesis is divided in five sections showing the natural process followed to reach the goal, starting with the gathering of information and knowledge about the theory and the related works, followed by the design of the disambiguation strategy and its implementation, then the strategy is evaluated using the selected research instruments, and finally the conclusion where the insight obtained from the results is presented. Details about each section can be read in the following list:

- **Background:** this chapter presents the necessary theory and concepts to develop a complete understanding of the following chapters of the thesis. The topics include Natural Language Processing (NLP), Attention and Transformers, LLMs and Prompting and finally Smart Systems and Agents as applications.
- **Literature Review:** here a literature review regarding smart systems is presented, the chapter is divided in three sections: From NLP to Action Composition and Beyond Action Composition. The order of the sections is arranged incrementally according with the semantic interpretation requirements of the goals.
- **Methodology:** the design and building process of the necessary instruments to achieve the thesis goal are presented here: Disambiguation Strategy Design, Strategy Implementation and Research Instruments.
- **Results:** in this chapter the research instruments application is presented together with the results and their respective analysis, it is divided in three sections; Interviews Analysis, Expriment Analysis and Prompting Performance.
- **Conclusion:** Here the highlights of the thesis are presented together with the report about the achievement of each objective.

Chapter 2

Background

2.1 Natural Language Processing (NLP)

Natural language processing is a machine learning technology that enables computers to interpret and extract qualitative aspects from a given input in natural language (the way humans express themselves) [1]. NLP has existed for decades, at the beginning most of the techniques were based solely on statistics and mathematics; specifically focused on word frequency, word embedding, similarity score, lemmatization and stemming.

In recent years most of the complex NLP tasks have been approached using Neural Networks, starting with Recurrent Neural Networks, followed by Long Short Term Networks and moving later to the top-notch Attention Techniques using Transformers, the technology behind ChatGPT. In this section the evolution of the NLP techniques and tasks is explored deeper as a way to prepare the reader for the incoming concepts regarding Large Language Models and Prompting.

2.1.1 NLP Tasks

A diverse array of techniques have emerged to tackle the multiple challenges faced by natural language processing, these challenges cover from extracting meaningful entities to generating coherent passages of text, NLP tasks play a pivotal role in enabling machines to comprehend and interact with human language.

Named Entity Recognition (NER) focuses on identifying and extracting specific entities or fields within a text corpus, such as names, dates, locations, organizations, and more. By providing the model with annotated examples of these entities, NER systems learn to recognize patterns and contextual hints, enabling them to accurately identify and classify instances of named entities within the text.

Text summarization aims to distill the main ideas and key points while preserving coherence and relevance. Using extractive methods, which select and assemble

important sentences or passages, or abstractive methods, which generate new sentences to capture the essence of the text.

Text Classification involves categorizing text documents or snippets based on predefined labels or classes, depending on the intended goal and the underlying structure of the text. Text classification can be supervised, utilizing labeled training data to train models to recognize patterns and make predictions, or unsupervised, where clustering techniques are employed to group similar documents based on their content or features. Applications of text classification range from sentiment analysis and spam detection to topic modeling and document organization.

Text Generation techniques enable models to produce human-like text based on given parameters and context, often leveraging deep learning models such as Recurrent Neural Networks (RNNs) or Transformers. By learning from a large corpora of text data, these models develop the ability to generate coherent and contextually relevant text, for a wide range of applications such as creative writing,automated content creation, chatbots or language translation.

2.1.2 NLP Techniques

The tasks above mentioned some of the techniques used to approach them, however a more complete list is shown below, it comprehends the traditional techniques which are interpretable and white-box by design and the black-box newer techniques that involve neural networks.

Word2vec embeddings aim to efficiently map words to dense vectors while preserving their contextual similarities. The main idea is to keep track of semantics distribution, thus words that occur in similar contexts are assumed to have similar meanings.

N-grams are sequences of N consecutive words in a text, where N can be any integer. These N-grams are used to model the probability of occurrence of certain word sequences in natural language, which is useful in applications such as language modeling, feature extraction in text processing, and detecting similarities between texts.

Recurrent Neural Networks (RNNs) are a type of neural network architecture specifically designed to process sequences of data, such as text, audio, or time series, where the relationship between sequence elements is the key. RNNs have feedback connections that allow them to maintain a kind of "memory" of what they have previously computed. This means that the output of one layer in an RNN is used as input in the next iteration of the calculation process, enabling RNNs to capture temporal patterns in the input data; this design makes them very effective for tasks such as language modeling.

RNNs may struggle to handle long-term dependencies that make the “memorized”

values vanish. To address this, variants such as *Long Short Term Memory (LSTM)* have emerged, incorporating flow control mechanisms to selectively learn and forget information over time, making them more effective in capturing long-term dependencies, increasing the capabilities to process long pieces of natural language text like blogs, chats and short books.

Even though these mechanisms can cover most of the simple tasks, when more generalization capacity is needed or more than one task is needed at the same time, neural networks can provide better results at the expense of less interpretability and more power consumption. The next section provides a brief introduction to neural networks on NLP and the basics of how they work.

2.2 Neural Networks for NLP

A Neural Network is a technique used for multiple tasks inside the machine learning realm, depending on the task the architecture of the neural network and the data pre-processing change, in the case of NLP related tasks there are specific architectures that have been developed over the years, in the following sections they are presented in chronological order.

2.2.1 Encoders and Decoders

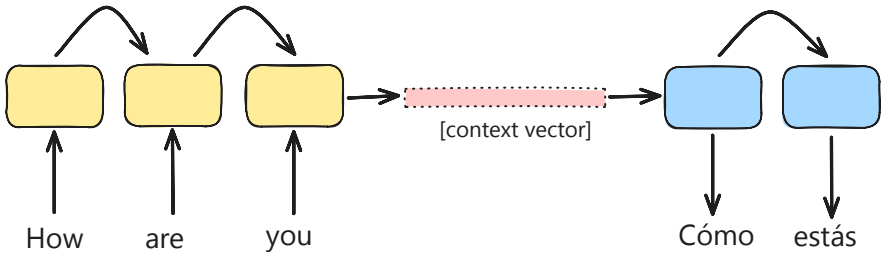

Figure 2.1: Simplified encoder-decoder architecture

Encoders and decoders are the backbone of sequential input, enabling them to tackle a wide range of sequence-to-sequence (Seq2Seq) tasks in natural language processing. The encoder processes the input sequence to create a meaningful context vector, while the decoder utilizes this vector to generate the output sequence token by token.

The Encoder primary function is to process the input sequence and extract its semantic interpretation into a single fixed-length vector, often referred to as a context vector, a compact representation of the sentence. The encoder

typically consists of layers of recurrent neural networks (RNNs), such as bidirectional recurrent neural networks (BRNN) or long-short term memory (LSTM) networks. These RNN layers process input tokens sequentially, updating their hidden states at each time step based on the current input token and the previous hidden state as shown in the figure 2.1. The encoder's final hidden state, which is the sentence representation, is used as the initial state for the decoder.

The Decoder is responsible for generating the output sequence based on this vector, the decoder typically includes layers of recurrent neural networks as well. At each step, the decoder uses its hidden state, starting with the encoder's context vector, together with previously generated tokens, to predict the next token in the output sequence. During inference, the decoder generates the output one token at a time, with each token influencing the generation of subsequent tokens until an end-of-sequence token is produced or a maximum length is reached.

This encoder-decoder architecture while powerful limits the text processing to a sequential paradigm, closing the possibility to parallelization and making escalation difficult. This problem was solved years later thanks to the introduction of transformers.

2.3 Transformers

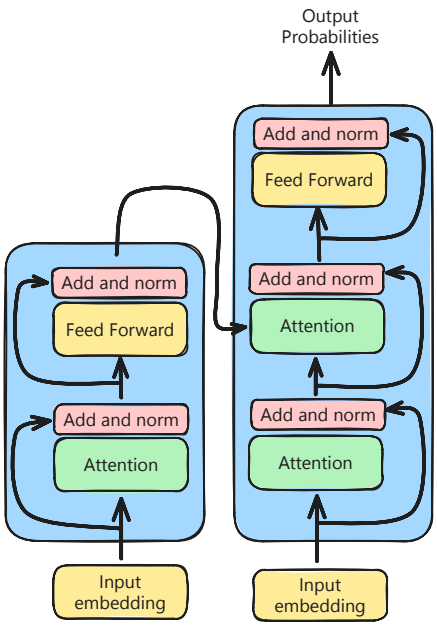

Figure 2.2: Simplified transformer architecture, encoder on the left, decoder on the right

Transformer architectures, which rely heavily on attention mechanisms, have revolutionized the field of NLP with their ability to capture long-range dependencies and parallelize computation across sequences. Transformers have become the backbone of many state-of-the-art language models, achieving remarkable performance on a wide range of tasks, including language translation, text generation, and language understanding.

The first transformer architecture was created in 2017 as part of Google Research [2], the main idea behind this architecture was removing RRNs from previous models and relying solely on attention blocks as shown in figure 2.2, thus the title "Attention is all you need". RNNs are useful for sequential data and remembering sequential patterns, however they have two main constraints, data processing cannot be parallelized and sample limits due to memory constraints.

Transformers on the other hand get the input embedding of the whole sentence at once, this thanks to not depending on RNNs, this innovation allows Transformers to scale better moving from the original one with six layers to models like GPT-3 with 96 layers.

The above work was pivotal in the NLP world, opening the door for better and more advanced models for multiple purposes, some of them increasing the barrier of single-task models, introducing fine-tuning where models are pre-trained for a general task and some additional short training iterations can be done to adjust it for a different one. The most known models are described below.

GPT, a private model released by OpenAI [3], was the first transformed based model after the creation of the technique; it is a decoder-only model that uses 12 layers of attention blocks. It brought forth the idea of using unsupervised learning for initial training and supervised learning for subsequent fine-tuning, a methodology widely adopted by multiple transformer based models. The strategy behind it was trying to guess the token to the right of the current one, in a process called masking, where an incomplete text is given and the model tries to fit the next token. BERT, formally "Bidirectional Encoder Representations from Transformers" [4], was the second transformer-based model, proposed by Google aiming to be used in its search engine. It is an encoder-only model, built stacking 12 encoders with attention blocks, in contrast to GPT BERT is bidirectional; it takes contextual information from both left and right tokens simultaneously.

2.4 Large Language Models and Prompting

Thanks to the increasing performance of language models like BERT and GPT, their scalable nature and the evolution of hardware to support them, the number of parameters in the neural networks started to increase year by year. These

large language models (LLMs) started to exhibit a behavior towards complying with specific tasks without fine tuning training, learning tasks without explicit supervision became a possibility [5]. This opened a door to a new world, where new concepts became relevant and new paradigms of fine-tuning language models raised.

Few-shot learning is a machine learning paradigm to train language models to recognize classes/tags that were not part of the training dataset providing few examples at inference time. Here the aim is to provide the model with enough generalization capacity to understand the semantics behind the training data, instead of just the attributes that characterize it. Doing this additional classes and information can be asked at inference time providing few examples and the model will provide a coherent response. For instance, suppose a model is trained to recognize mammals and reptiles from images, but it has never seen birds during training; if the model is a few-shot learner, just 2 examples of birds are provided at inference time and it can use these examples together with the learned features of mammals and reptiles to know what is a bird and what is not, even though it hasn't seen them in the training phase.

Zero-shot learning is a machine learning paradigm to train language models to recognize classes/tags that were not part of the training dataset providing zero examples at inference time. By definition, zero-shot learners are more powerful than few-shot learners; here the model should be able to complete the task using classes that it has never seen before. Taking the same example; if the model is a zero-shot learner, it can use the learned features from mammals and reptiles to classify birds, without any information about them. Zero-shot models need more background and context at training time than few-shot learners because they should use it to infer new knowledge without any examples.

LLMs were discovered to be few-shot learners and zero-shot learners when fine-tuned [6], this started a new paradigm for using language models, a huge amount of resources can be saved by using pre fined-tuned models and relying on few-shot learning at inference time to customize it for specific purposes. The way of asking, solving tasks and providing examples to language models at inference time is now known as prompting.

2.4.1 Structure of a Prompt

A prompt has the following elements:

- Instruction: A precise task or directive for the model to execute.
- Output Indicator: How the model should return the output

- Context: Contextual information used by the model to accomplish the task.

Example:

- *Instruction:* Translate the given english text to spanish.
- *Context:* "I love Large Language Models and Machine Learning"
- *Output Indicator:* Return the translation using only lower-case letters.
- *Expected response*: *yo amo los modelos de lenguaje grandes y el aprendizaje automático.*

2.4.2 Prompting Techniques

Throughout the years multiple prompting techniques have been created to solve multiple tasks [7], the most popular techniques are shown below. The examples are shown with the prompt marked as ***P*** and the response marked as ***R***.

Zero-shot: As defined above, zero-shot prompting works by asking the model for a task without any additional context. The model should be capable of understanding it even though no example of this was given in the training process.

P: Reverse the following sentence: Torino mi mancherà
R: `mancherà mi Torino`

Few-shot: Few-shot prompting works by asking the model for a task providing few examples of how to do it (Language Models are Few-Shot Learners). Taking the previous example, by default the model reverse words, providing an example we can change the task to do it character-wise:

P: Reverse the following sentence: L'Italia è bella
R: `alleb è ailatI'L`
P: Reverse the following sentence: Torino mi mancherà
R: `àreahcnam im oniroT`

Chain-of-thought (CoT): Introduced by (Chain-of-Thought Prompting Elicits Reasoning in Large Language Models), it aims to increase the model capacity to reason, providing an example not only of the answer but the thought flow as well. This way the model can response more accurate answers. To continue with the previous example we add to the task the removal of the vowels, if asking without chain-of-tought the model fails, however when the thought flow is provided, the answer is correct:

P:
Reverse the following sentence character by character, if the character is a vowel, remove it: L'Italia è bella.
To do it first remove all the vowels from the sentence: L'tl bll
Then reverse the resulting string: llb lt'L
Reverse the following sentence character by character, if the character is a vowel, remove it: Torino mi mancherà
R:
`Removing all the vowels from the sentence: Trn m anchr Reversing the resulting string: rhcna m nrT`

Prompt chaining: Prompts can be chained together to achieve multi task processes; the response of one prompt is injected in the next one to make the model complete a complex task without the reasoning requirements.

Other methods based on templating and prompt optimization have been developed, focused on using ML techniques to optimize the prompt for specific tasks and language models, some of them are Tree of Thoughts [8], Self-consistency [9] and Retrieval Augmented generation [10].

Chapter 3

Related Works

3.1 From NLP to Action Composition

The related works in this section are focused on the user interpretation and action composition. Most of them try to improve the quality of the possible action or rules suggested to the user, based on the available devices, the context and the user input. The level of ambiguity managed here is focused on the same three aspects as well, however in some of them the interest in user feedback is better, trying to align with a more user-centered system.

Corno et al. propose *EUPont* [11], an ontology based on the semantics of the entities involved in Smart home systems; it enables the development of advanced IoT applications capable of adjusting to multiple contextual scenarios. Using the ontology inside an EUD (End-User Development) environment the interpretation of the user intent and the composition of action over the devices is highly modular.

RecRules[12] is a recommender system for End-User Development (EUD) tools. It suggests IF-THEN rules based on functionality, not brands, by using semantic reasoning. The power behind it is a graph based semantic network, over which reasoning is possible to recommend the best set of trigger-action rules to the user.

The system is focused on the complete smart-home flow, covering user intent interpretation, command composition and command-device mapping. Additionally RecRules is context-aware, allowing personalization depending on the individual use of the system, this thanks to the path recording capabilities that can be implemented over the network.

Ambiguity is managed statically in this system, when the graph is created all devices, actions and parameters are mapped to specific semantics, creating a limited space for the system to operate in. However the ambiguities present in the user input are not taken into account.

Similar to RecRules, Corno et al. propose again a Conversational Search and

Recommendation (CSR) system[11] able to suggest pertinent IF-THEN rules. The system is context aware since the logical process followed involves three main steps; interpreting the current user's intention, understanding the available devices and using the context and the user input to create custom recommendations.

One highlighted aspect of this system is the capacity to adapt and solve ambiguities dynamically, asking for user feedback at running time when none of the suggestions is taken by the user or additional parameters are required, this clearly shows a more user-centered approach, as it tries to get what is the actual intent.

Gallo et al. introduce a technique to construct a conversational agent aimed for smart environments[13]. This approach combines ChatGPT and Rasa, enabling the creation of trigger-action rules and management of smart devices. It uses ChatGPT's capabilities in generating open-domain dialogues and leveraging Rasa's functionalities for intent, entity, and action handling.

It is mainly focused on the creation of trigger action rules, Rasa is used for NER and dialogue flow tasks and the LLM is used for simplification purposes, using its semantic interpretation to reduce complex rules into simple commands. Additionally, the language model is used for interpretation purposes, solving questions regarding the context and the devices.

The authors try to achieve a complete dialogue flow using natural language while creating accurate and coherent trigger-action rules. Regarding ambiguity managing, the system manages it manually, with an internal loop asking for additional parameters or clarifications before proceeding to command execution on the device.

Wang et al. note that while IFTTT is easy for non-programmers to use, its simple nature often holds it back from handling complex rules and conditions needed in real-world scenarios. To tackle this problem, an scripting approach is proposed[14], where users can program different dynamics across devices and save states and values that can be reused to interact with them.

This work is focused in the expressiveness and power of simple trigger-action paradigm, as it intends to extend the possibilities of smart systems, the user does not play a central role. The easiness of use of the system or the improvement of the user-system interaction is not a big concern for them. However the interesting part of this work for this thesis is the aim of getting out of the traditional IFTTT paradigm and open the door to a wider range of possibilities.

3.2 Beyond Action Composition

Some related works go beyond the action composition and try to fulfill larger goals, some of them are able to use external tools to do it and others are focused on interpreting a complex input for the user. They have in common that they use

LLMs to achieve it, the generalization power of these language models allows the authors to pursue bigger goals and improve the interaction with the user, going closer to the real natural language interaction.

Toolformer[15] is a system that is able to use external tools through their APIs, decide which of them to call depending on the necessities, and what arguments are needed in the request. The set of tools include a calculator, a QA system, a search engine, a translation system and finally a calendar. The approach they took was to fine-tune a language model including API calls and their responses inside some training prompts, to make it able to recognize when the API should be used and the response format.

Different experiments were conducted showing that the 6.7B parameters model GPT-J is capable of using tools exceeding the performance of much larger models like GPT-3 in all the tasks, setting an alternative when the set of tools to use is constant and the available resources are limited.

Similar to Toolformer, *OpenAGI*[16] intends to use multiple tools to achieve a specific goal, however it goes even further, trying to use the tool to achieve a bigger and abstract goal, the strategy they follow is to identify the useful tools, create a detailed plan, execute the plan and parse each result to feed the following step in the plan. This combination of steps allow the system to get closer to a general AI, without the necessity of fine-tuning and only relying on few-shot prompting the system is able to comply with zero-shot goals.

Sasha[17] is a system that follows the same idea of OpenAGI and Toolformer, however it is entirely focused on smart-home systems. The main goal of it is interpreting implicit goals given by the user, using the power of language models to remove ambiguities from the user input and replace them with concrete concepts that have no ambiguities in the context. Even though disambiguation is one its main goals, it is done using the own definition given by the model.

Chapter 4

Methodology

The smart word inside the SHS expresses the capability of the system to interpret the user intent and compose an action to be executed over a set of devices. This interpretability depends on the context, the user, and the intent itself. An intent can be ambiguous in a context and can be clear in another, it can be ambiguous depending on the specificity (as defined by Sasha) or can be ambiguous depending on the physical devices available.

To disambiguate the user intent is necessary to know first what is an ambiguity in the context of this thesis; an ambiguity is defined as a concept or term inside the user intent that can be interpreted differently by two or more people within the given context. This definition is taken since it is concrete and has a criteria to decide whether or not something is ambiguous, it is measurable knowing the number of people that interpret the ambiguity differently and it is, and thus is comparable as well having the option to have ambiguities that are more ambiguous than others.

Knowing the definition is time to proceed with the next question in the queue; how can it be disambiguate? This question is solved in the following subsections following 5 basic principles to make the system user-centered.

4.1 The 5 Principles

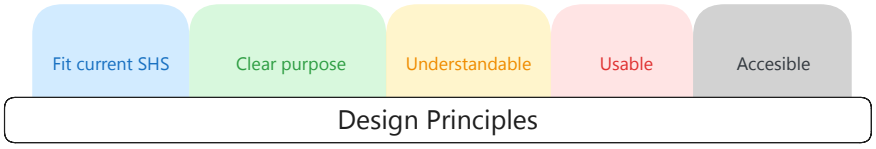

Figure 4.1: Design principles

Systems presented in the related works chapter made an attempt to solve this, attacking specific parts of the problem, however none of them approach the user to solve it. The methodology follows 5 principles based on the well known UX principles[18] as shown in the figure 4.1; each step of the design and implementation process is developed on top of them.

4.2 Disambiguation Strategy Design

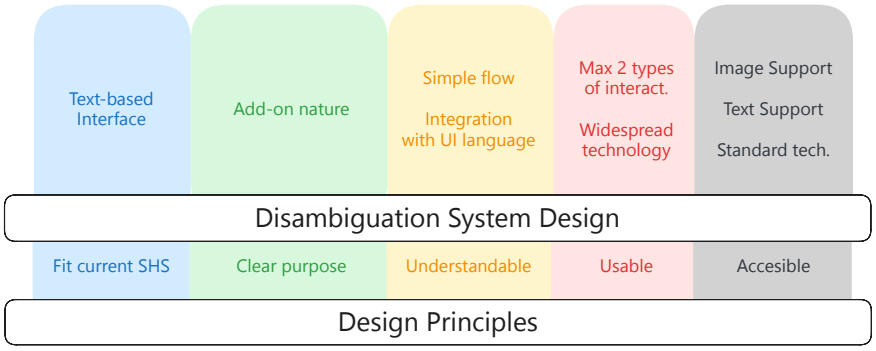

Figure 4.2: System design actions in line with design principles

To design the disambiguation strategy each principle is mapped into concrete design actions:

The *Fit current SHS* principle is implemented making it text based as most SHS with a graphical user interface, it should support images and audio as resources to make it more expressive.

To have a *Clear purpose* it is delivered as an add-on or extension on a conventional text system, the user interacts with it as part of the underlying system not as a standalone application.

To be *Understandable* the disambiguation system should have a simple flow, not more than two branches and avoid keeping states. Additionally it should integrate seamlessly with the UI language, using the same design principles.

Making it *Usable* requires using a widespread technology that is easy to access and use by the user, at the same time the interaction should use a maximum of two input methods, to avoid increasing the effort of using the system.

Finally, *Accessibility* is included using a standard technology that is already supported by most of the OS accessibility features like iOS voice to text, Android's dictation, etc. Additionally offering the option of using text or images.

4.2.1 The Design

To disambiguate a concept or term multiple resources can be used, following the accessibility and usability principles, even though the system is text based, disambiguation options should not be limited to one type of resource; following this images are included together with text and configured based on user preferences.

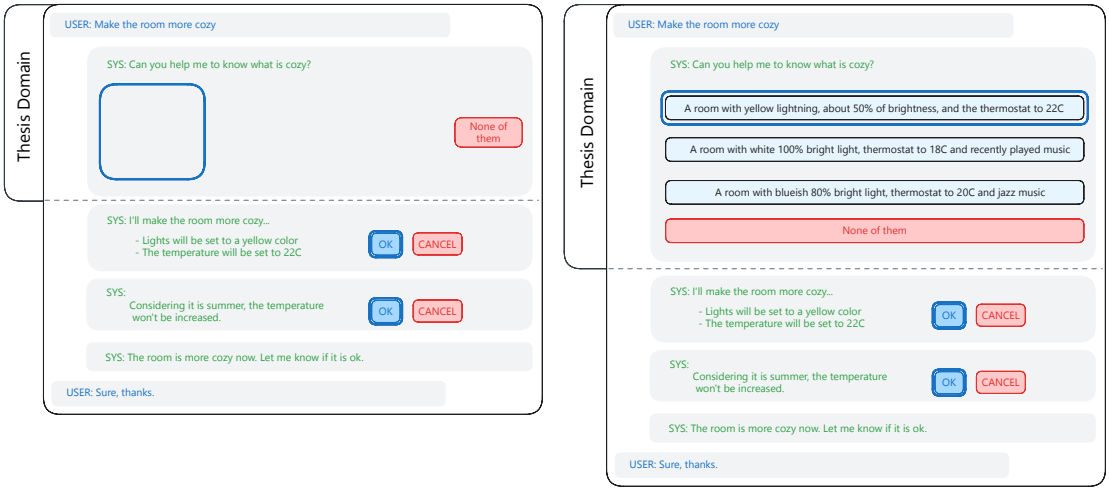

Figure 4.3: Text-based design and flow when an ambiguity is found. (image options on the left, text options on the right)

The design actions are materialized in a chat-like system. The disambiguation flow is started whenever the system detects an ambiguous term inside the user intent, then based on the user preference (text or images) disambiguation options are shown, in the figure 4.3 it can be seen how the system should behave when an ambiguity is present. Given the options the user can select one of them, then the disambiguation system should be capable of translating that selection into useful information and send it together with the original intent to the underlying SHS.

The input method is button based where the user selects with only one click the desired option. It has only one flow, the user selects an option or none of them, nothing else is needed. The rest of the flow is not in the domain of the thesis and is controlled by the SHS.

Passive features not shown in the image are designed as well; the system takes into account the existing devices and the user context to show the disambiguation options, previous interactions are taken into account as well, the system saves the disambiguate concepts to use them in the future to avoid recurring to the user to often and ruin the interaction experience.

4.3 Disambiguation Implementation

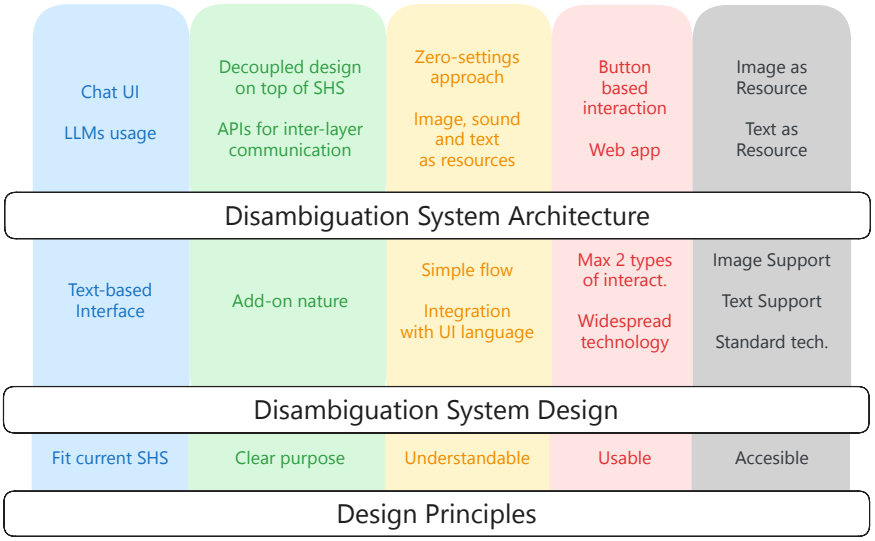

Figure 4.4: System architectural decisions based on design principles.

To implement the designed disambiguation strategy it should be supported by a backend architecture that make it alive, following the same principles again multiple decisions are taken:

- The text based interface uses a Chat UI embedded inside a web engine (browser, device, etc) and LLMs to power the disambiguation add-on.
- The disambiguation does not depend on the underlying system, to make it an add-on, it is decoupled from it and HTTPs APIs are used as inter-layer communication.
- The simple flow is kept using a zero-settings approach, and using a unique UI components library.
- The interaction type is decided to be button based and text based.
- Third party services to translate image to text and text to image should be included.

4.3.1 System Architecture

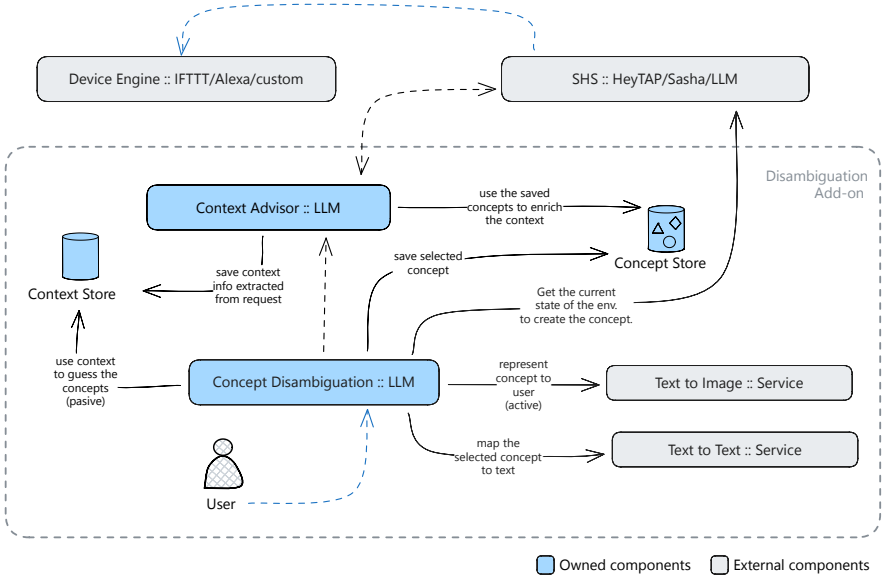

Figure 4.5: System backend architecture.

The system architecture is designed to leverage the power of LLMs while keeping the design principles in line. Multiple components are integrated to materialize the features created in the design phase as shown in the figure 4.5.

Concept Disambiguation is the main component of the system, it is a component powered by GPT-4 as language model, it contains the necessary prompts to detect disambiguations and make the proper requests to other components. Additionally, it saves user selected concepts to be used in the future.

Context Advisor is in charge of saving useful information about the context and the devices using the already stored disambiguations, it depends on the information provided by the external SHS.

Text to Image and *Text to Text* are external services to get images and text fragments to disambiguate a specific term or concept.

Language Mapper is the actual SHS in charge of mapping the NLP input to the device action, the disambiguation system runs on top of them and it is out of the scope of the thesis to develop this system, however as it is a dependency an external one should be used or simulated, examples of this can be HeyTAP, Sasha, EUPont, etc.

The *Smart Engine* is the last link in the chain, it is the actual engine that expose the protocol to use the devices, Google Home, Alexa, Zigbee, etc are examples of them.

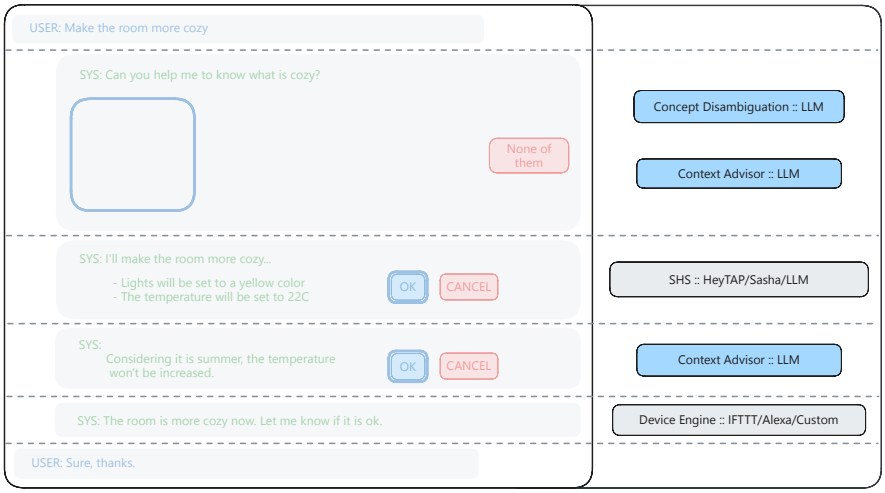

Figure 4.6: System architecture components present in each user interaction step.

The user interaction is reflected in the figure 4.6, where each component is mapped to its participation inside the user interaction flow. It starts when the user sends an intent to the system where the first component catches it, the disambiguation system, it tries to find ambiguities in the intent, if nothing is found the request is redirected to the SHS, otherwise the options are generated using the resources services enriched by the context advisor and returned to the user to expect the selection, once the selection is made, the resource is translated to text and the original intent is complemented to disambiguate it before being redirected to the underlying SHS. This system returns the action to take and it can be changed before proceeding based on the context advisor's input.

4.3.2 System Implementation

Due to time and resources constraints the complete architecture is not implemented at all, the Context Advisor and the memory feature represented by the Context Store and the Concept Store are not implemented as shown in figure 4.7, however the main flow is kept with Concept Disambiguation. The components are implemented using GPT-4 as the main language model, not only to power the necessary features but to simulate external systems, DALLE-2 is used as the Text to Image component with the minimum resolution.

The chat UI is implemented using the Azure Chat Template together with the Microsoft design language, the UI components were designed as close as possible to match it, at least in behavior, the following figure shows an image of the mentioned UI.

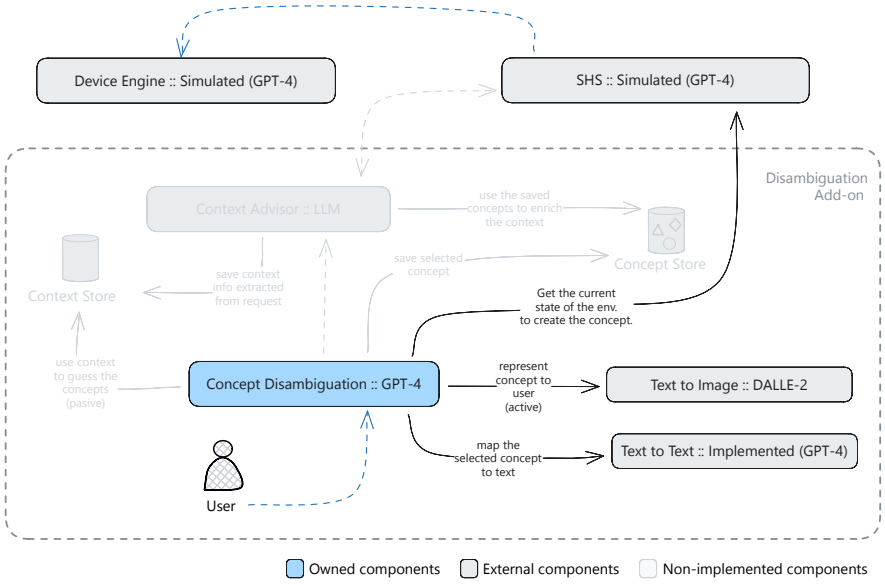

Figure 4.7: Implemented components from the system architecture.

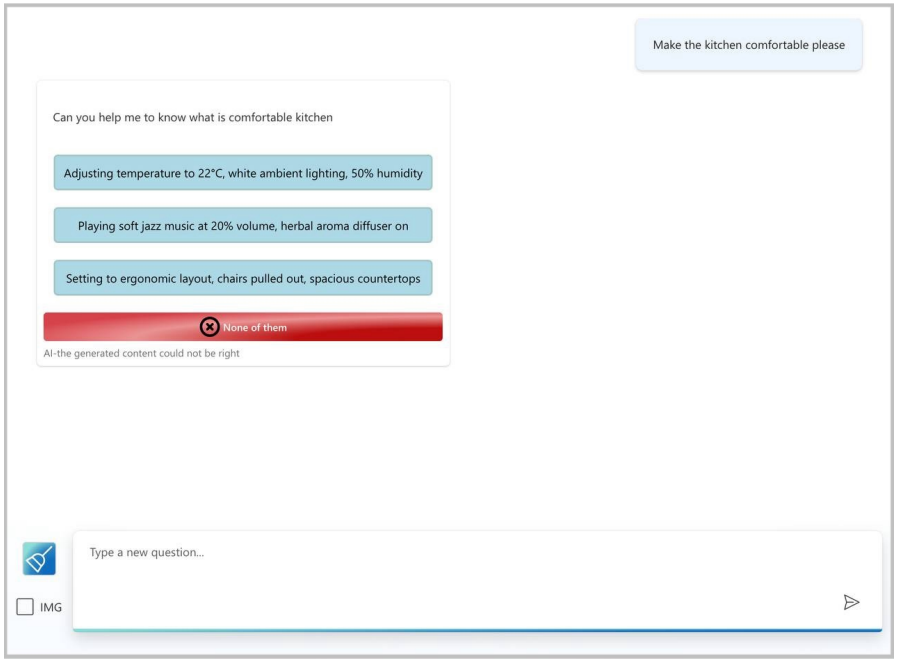

Figure 4.8: Disambiguation system implementation.

4.3.3 Prompting Strategies

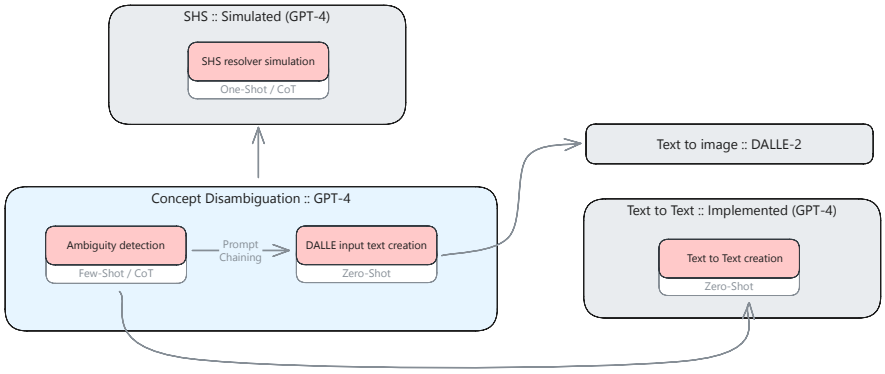

Figure 4.9: Prompts and prompting strategies per component.

As shown in figure 4.9, multiple components use different prompting strategies depending on the task complexity and importance. The most important prompt Ambiguity Detection is the one that uses the most complex prompting strategy, to establish the proper prompting technique a test set of 45 mock intents tagged with the resource type that should be selected and the present ambiguity is created, one fragment is show in the listing 4.1. Zero-Shot, Few-Shot and Act-As prompting techniques are tested over the data set, setting the best performer.

```
- instruction: Play a random podcast
  ambiguities: []
  resource_type: audio
- instruction: Adjust the speaker volume for a vibrant
               living room environment
  ambiguities:
    - vibrant
    - vibrant environment
  resource_type: audio
```

Listing 4.1: Test intent sample

For the rest of the instruments the minimum prompting technique that offers an appropriate performance was selected to optimize resources consumption:

- SHS resolver simulation: to simulate the SHS system the most simple technique was used since the simulated system result do not affect the conducted experiments.
- DALLE input text creation: here One-Shot is used, one example of the requesting format is necessary to make it work properly.

- Text to text creation: Zero-Shot is enough for the model to create the text options, thanks to the injected environment, and the original user instruction.

4.4 Research Instruments

4.4.1 Interviews

Two interview sessions were designed by Tomaso Calo as an advisor to know multiple aspects related to the interviewees' pre and post experiment opinions. The interviews are structured, all the questions are the same for all the participants.

Pre-Experiment Interview		
Category/Question		**Goal**
Background Information		Know the familiarity of the users with similar systems
1	How familiar are you with using smart home systems or devices?	
2	Have you previously used voice or text-based assistants (like Alexa, Siri, etc.)?	
Expectations and Preferences		Understand what resource the interviewee prefers based on previous experiences between text and images
3	What are your expectations from a smart home system in terms of ease of use, responsiveness, and accuracy?	
4	Do you have a preference for text-based or image-based interactions? Why?	
Conceptual Understanding		Know how the interviewee interacts with smart systems
5	How do you usually convey your needs or commands to a smart device?	
6	Can you provide an example of a situation where you found it challenging to communicate your intent to a smart home system?	

Table 4.1: Pre-experiment interview design

Post-Experiment Interview	
Category/Question	**Goal**
Satisfaction and Alignment	Know alignment between expectation and system's results
1	How well did the system's response (either text-based or image-based) align with your initial interpretation of the command?
2	Were there any particular aspects of the response that you found particularly effective or ineffective?
Ease of Use and Preference	Know how usable was the system and how close it was to the design principles
3	Did you find the provided modality (text or images) intuitive and easy to use?
4	Given the choice, would you prefer to use the modality you tested in your daily interactions with a smart home system? Why or why not?
Comparative Perception	Understand what resource the interviewee prefers based on the current experience between text and images
5	Which modality do you believe would be more effective in different scenarios or contexts?
Suggestions and Improvements	Get feedback about positive and negative aspects of the system
6	Are there any changes or improvements you would suggest for the modality you interacted with?
7	Can you think of any specific scenarios where this modality would be particularly beneficial or problematic?

Table 4.2: Post-experiment interview design

4.4.2 Experiments

To conduct the experiment the first requirement is to have a smart home, in this case that environment was simulated. The following figure 4.10 was presented to the users as the smart home where they would be inhabitants during the experiments.

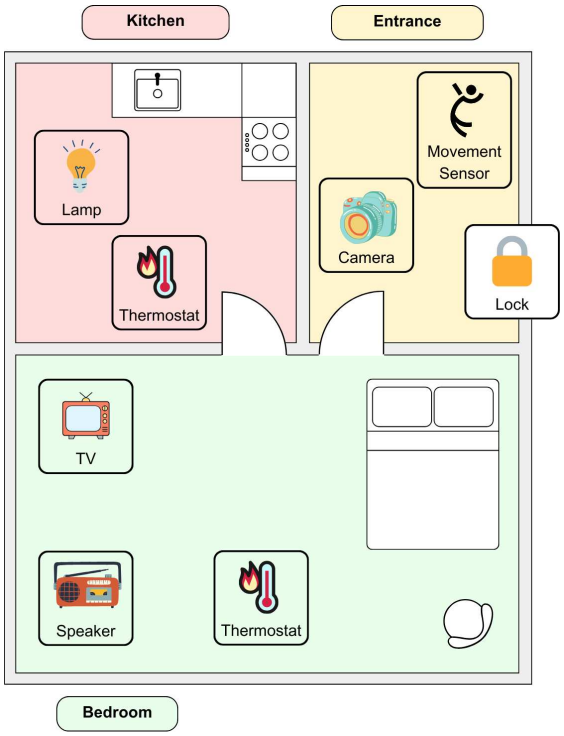

Figure 4.10: Simulated smart home.

The experiments were set in a way that user experience can be compared; 4 intents are selected to be presented to the user one by one, for each intent the user is asked about the expectation of the command's result, then the disambiguation options for the intent are presented, randomly selecting between images or text, and the user scores them, selects one and comments why that option was selected and why the score was given. Finally the system result is scored as well and additional comments can be collected if desired. The table 4.3 presents the experiment schema for one intent, all the used intents are presented in the listing 4.2.

Field Name	Description
Modality	Text or Image
Command	One of the 4 intents
Expectation	The user expectation about the command
Disambiguation Options	Disambiguation options generated by the system
Selected Option	The option selected by the user
Score	A score from 1 to 5 where 1 is not satisfied with the options and not aligned with the expectation. 5 means the opposite.
Comment (Optional)	Why that option was selected and why the score was given
System's Result	The simulated SHS result
Score	A score from 1 to 5 where 1 is not satisfied with the result and not aligned with the expectation. 5 means the opposite.
Comment (Optional)	Why that score was given

Table 4.3: Experiment schema.

```
intents:
 - instruction: Set romantic lights in the kitchen
 - instruction: Create a relaxing ambiance when I arrive home
 - instruction: Turn on the lights if a child enter the house
 - instruction: Make the kitchen cozy
```

Listing 4.2: Intetns used in the experiment.

Chapter 5

Results

5.1 Sample

Seven people were selected to apply the instruments on, the age was specifically selected for them to be the most familiar as possible with Smart Agents.

#	Age	Nationality	Gender	Interview language
1	24	Colombia	F	Spanish
2	27	Colombia	M	English
3	25	Colombia	F	English
4	29	Colombia	F	Spanish
5	13	Colombia	M	Spanish
6	34	Colombia	F	Spanish
7	28	Colombia	M	Spanish

5.2 Interviews Analysis

5.2.1 Pre-Experiment

While most interviewees are familiar with smart systems from user or engineering perspectives, none of them have experienced smart systems integrated with Internet of Things (IoT) devices. Six out of seven respondents have used smart assistants mainly for simple tasks such as jokes, searches, and message composition, but **none have employed them for more demanding tasks**.

Regarding expectations from a SHS, participants highlight the importance of conveying commands naturally and processing intents without excessive specificity. Response **coherence and consistency are more important for them than**

high accuracy, with users prioritizing human-like responses. Other factors include low configuration, ethical data management, and up-to-date information. Regarding format preference, users appreciate text for its explicitness, while users value images because of the volume of information transmitted. Overall, there is no clear bias towards a format.

The predominant methods for composing intents for smart assistants are through key words and specific details. However, most participants claim having issues using them, with some intents being entirely misunderstood due to a lack of context, while others are only partially recognized, reflecting this in unexpected results, for instance the experience of interviewee #1: "Once I wanted the device to tell me what the weather would be like the next day, –Weather Bogotá Tomorrow–, it started to respond and halfway through the response it went silent, and played a song on Spotify".

5.2.2 Post-Experiment

Users find the system's responses aligned well with their expectations, nevertheless, sometimes needing minor adjustments. According to them, **responses are generally close to expectations, even if disambiguation options were occasionally confusing**. Image-based responses were favored for better alignment compared to text-based ones. Effective aspects included response time, intuitiveness, and device awareness. Users appreciated clarifying questions and image-based responses for better comprehension. However, some found **issues with temperature options** (setting the same one for warm and cold) and **limitations in image variety and size**.

Preferences for interacting with smart home systems after the experiment varies among users, with some favoring images even with their limitations in capturing all aspects of their thoughts, while others preferred text for its precision, the individual preference does not change after the experiment. There's a preference for images among those who find reading tedious, as **they believe images offer a quicker way to understand information**. Overall, **the consensus leans towards images** being more effective in conveying information.

Suggestions for improvement include:

- Providing descriptions alongside images to enhance understanding.
- Allowing the selection of both text and image when choices complement each other and enabling the selection of two or more options for more accuracy.
- Users emphasized the importance of aesthetic appeal in images, recommending visually pleasing and contextually appropriate visuals.

In general, while some participants highlighted the limitations of images in representing certain aspects like sound and temperature, others found them beneficial for conveying ideas, particularly for those grappling with abstract concepts. **Text was noted for its clarity, but images were recognized for their effectiveness in conveying references**.

5.3 Experiments Analysis

Intent 1

Intent: Create a relaxing ambiance when I arrive home.

Users appreciate the well-described options but emphasize the importance of sound for relaxation. They generally like warm lighting and jazz music, with a preference for lower temperatures. **Some express interest in selecting multiple options**, since the definition of relaxation can contain more than one aspect and usually involves many of them like sound, temperature, lightning, smell, etc.

Regarding the system response, users agree that the system effectively captures various aspects of ambiance but lacks consideration for sound. Temperature adjustments are suggested, with **one user initially disagreeing but later changing their mind after seeing the suggested disambiguate options. Another user appreciates an unexpected positive change made by the system** regarding closing the entrance.

Intent 2

Intent: Make the kitchen cozy

Users find the environment depicted in the images close to their expectations. Some appreciate the accurate lighting but express that temperature cannot be conveyed visually. Preferences include dimmed lights, low-volume music, and coffee for a cozy atmosphere. However, some users feel **the image doesn't fully meet their expectations, lacking temperature indication and additional options.**

In the system's result, users generally appreciate the consideration given to lighting but express disappointment in the lack of temperature control, particularly in a kitchen setting where it's important. Some note **discrepancies in color accuracy between disambiguation options and the result**. While efforts are acknowledged, dissatisfaction remains over temperature settings.

Intent 3

Intent: Set romantic lights in the kitchen

Users find the textual option more descriptive and closer to their imagination. Some express disappointment with images lacking a kitchen setting and criticize harsh or overly orange lighting, preferring a cooler and dimmer ambiance for a romantic atmosphere. Despite differing opinions on ideal lighting, **there's consensus that disambiguation options and dimmed lighting contribute to a romantic feel.**

Users find the system's result satisfactory, with varying opinions on brightness and color adjustments. Some appreciate the ability to change light color. However, concerns are raised about unexpected/crazy results like pulsating light. Overall, while it meets expectations for some, improvements are desired for others.

Intent 4

Intent: Turn on the lights if a child enter the house

Users have mixed opinions about the provided options for the age limit. Some feel they are close to expectations but suggest considering additional factors. Others find none of the options fully align with their expectations. **There's debate about the representation of a child's height and age**. Despite disagreements, users generally find the options satisfactory, with **one user expressing difficulty in generating more ideas for categorizing a child**. They are glad that some options they didn't think about are useful.

Users have mixed opinions on the provided result. While some find it aligned with their expectations, others feel it lacks important details such as specifying the importance of the person's height or light color. Concerns are raised about **how the system detects a child and discrepancies in the age limit**. However, **one user appreciates the honesty of the response.**

5.3.1 Score Analysis

	Avg. Score for Disambiguation Options	**Avg. Score for SHS's results**
Intent 1	3.79	3.99
Intent 2	4.19	3.07
Intent 3	4.26	3.93
Intent 4	4.21	3.76
Intent Average	4.11	3.69

Table 5.1: Prompting techniques results on the creation of the disambiguation prompt.

Something noticeable is the lower scores on the system's result column compared

to the disambiguation options, since the environment is simulated the performance of the SHS system is not evaluated here and is out of the scope of the thesis. However, interesting aspects about the user can be extracted from it in the qualitative analysis.

The scores align well with the qualitative analysis of the experiment; the outstanding low score for the disambiguation options in the first intent is caused by the wide range of device action that can make an environment relaxing, the changes and options are too many that **the UI with single selection is not able to meet the user expectations.**

Intent 3 has the better score for disambiguation options, it is explained by the reduced range of options to fulfill the user expectation. This intent specifies a device out of the two in the room, making it possible for the user to imagine options within the scope of only one device in comparison with the intent 1. **The disambiguation system not only depends on the specificity of the request but the range of action of the intent.**

5.4 Prompting Performance

The creation process of the disambiguation prompt involved the testing of three prompting techniques; Zero-Shot, Few-Shot an Act-as, the performance results for each one are show in the table 5.2

Prompting Technique	TP	FP	TN	FN	Failures	Precision	Recall
Zero-Shot	31	43	1	0	3	0.419	1
Few-Shot	28	24	1	0	0	0.538	1
Act-as	27	24	7	6	0	0.529	0.8

Table 5.2: Prompting techniques results on the creation of the disambiguation prompt.

Precision across the three prompting techniques does not vary that much, moving around 0.5, this indicates a medium-low quality in the classification process; even though the detection of ambiguities is correct (high TP) thanks to the GPT-4 horse-power, its ability to know when something is not ambiguous should be refined (FP almost as high as TP). The lowest precision value is Zero-Shot prompting; the lack of proper examples increased the number of false positives ambiguities, in the other hand it indicates an improvement when examples are given and even more when the reasoning behind it is provided as in the Few-Shot prompting using CoT.

Recall is high in all the prompting techniques, indicating that the relevant ambiguities are being detected, **the system is biased toward detecting them**,

is over-sensitive since almost no false negatives where returned in the first two techniques.

The above could represent a problem for the inclusion of LLMs in disambiguation strategies since the user-system interaction can be armed due to the high number of false positive ambiguities. Corrections to the prompting straegy should be done to increase the precision even at the cost of decreasing the recall; **FP are more harmful than FN.**

Chapter 6

Conclusion

Through pre and post-experiment interviews, it was evident that users prioritize response coherence and consistency over high accuracy in SHS interactions, especially when engaging in more demanding tasks, if the response is appropriate for the context they are not concerned about fulfilling their own expectations, and sometimes consider valuable new ideas they did not think about. Regarding the system responses, some users encountered issues with disambiguation options, particularly finding them occasionally confusing. While text was noted for its clarity, images were recognized for their effectiveness in conveying references, leading to a consensus leaning towards the preference for images in conveying information when too much details are not needed.

Notably, users would appreciate the flexibility to select multiple options since some ambiguities involve multiple devices and properties. Additionally, users appreciate the efforts made by the system to meet their expectations, even though some discrepancies and limitations were noted, especially in hard-to-communicate properties like temperature or noise, besides that and image variety and more accurate visual representations is advised.

The prompting performance analysis revealed insights into the efficiency and biases of different prompting techniques, highlighting the importance of balancing precision and recall to optimize user-system interaction. The over-sensitivity of the system in detecting ambiguities, primarily due to false positive ambiguities, suggests a need for refining the prompting strategy to enhance precision, as false positives can potentially disrupt the user experience more than false negatives; users do not want to be constantly interrupted to be asked about an ambiguity that does not exist.

Finally, the introduction of a disambiguation system is highlighted as positive by the users, since it closes the gap between the system and the user increasing the coherence and consistency in the SHS responses, being something considered as valuable by the user. The technological infrastructure is able to implement this kind

of systems, however more advancements regarding prompting design and language model customization is needed to make it usable in a production environment.

Bibliography

[1] Salvatore Fanni, Maria Febi, Gayane Aghakhanyan, and Emanuele Neri. «Natural Language Processing». In: Sept. 2023, pp. 87–99. ISBN: 978-3-031-25927-2. DOI: 10.1007/978-3-031-25928-9_5 (cit. on p. 4).

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762[cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 03/10/2024) (cit. on p. 8).

[3] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. «Improving Language Understanding by Generative Pre-Training». In: () (cit. on p. 8).

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. May 24, 2019. arXiv: 1810.04805[cs]. URL: http://arxiv.org/abs/1810.04805 (visited on 03/10/2024) (cit. on p. 8).

[5] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. «Language Models are Unsupervised Multitask Learners». In: () (cit. on p. 9).

[6] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. *Finetuned Language Models Are Zero-Shot Learners*. Issue: arXiv:2109.01652. Feb. 8, 2022. arXiv: 2109.01652[cs]. URL: http://arxiv.org/abs/2109.01652 (visited on 06/25/2023) (cit. on p. 9).

[7] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. «Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing». In: *ACM Computing Surveys* 55.9 (Jan. 16, 2023). Number: 9, 195:1–195:35. ISSN: 0360-0300. DOI: 10.1145/3560815. URL: https://dl.acm.org/doi/10.1145/3560815 (visited on 06/23/2023) (cit. on p. 10).

[8] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. Dec. 3, 2023. DOI: 10.48550/arXiv.2305.10601. arXiv: 2305.10601[cs]. URL: http://arxiv.org/abs/2305.10601 (visited on 03/25/2024) (cit. on p. 11).

[9] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. Mar. 7, 2023. DOI: 10.48550/arXiv.2203.11171. arXiv: 2203.11171[cs]. URL: http://arxiv.org/abs/2203.11171 (visited on 03/25/2024) (cit. on p. 11).

[10] *Retrieval Augmented Generation: Streamlining the creation of intelligent natural language processing models*. URL: https://ai.meta.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/ (visited on 03/25/2024) (cit. on p. 11).

[11] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. «A high-level semantic approach to End-User Development in the Internet of Things». In: *International Journal of Human-Computer Studies* 125 (May 2019), pp. 41–54. ISSN: 10715819. DOI: 10.1016/j.ijhcs.2018.12.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S1071581918301228 (visited on 08/17/2023) (cit. on pp. 12, 13).

[12] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. «RecRules: Recommending IF-THEN Rules for End-User Development». In: *ACM Transactions on Intelligent Systems and Technology* 10.5 (Sept. 30, 2019). Number: 5, pp. 1–27. ISSN: 2157-6904, 2157-6912. DOI: 10.1145/3344211. URL: https://dl.acm.org/doi/10.1145/3344211 (visited on 08/14/2023) (cit. on p. 12).

[13] Simone Gallo, Alessio Malizia, and Fabio Paternò. «Towards a Chatbot for Creating Trigger-Action Rules based on ChatGPT and Rasa». In: () (cit. on p. 13).

[14] Marx Boyuan Wang, Daniel Manesh, Ruipu Hu, and Sang Won Lee. «iThem: Programming Internet of Things Beyond Trigger-Action Pattern». In: *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. UIST '22: The 35th Annual ACM Symposium on User Interface Software and Technology. Bend OR USA: ACM, Oct. 29, 2022, pp. 1–5. ISBN: 978-1-4503-9321-8. DOI: 10.1145/3526114.3558776. URL: https://dl.acm.org/doi/10.1145/3526114.3558776 (visited on 07/21/2023) (cit. on p. 13).

[15] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. *Toolformer: Language Models Can Teach Themselves to Use Tools*. Issue: arXiv:2302.04761. Feb. 9, 2023. DOI: 10.48550/arXiv.2302.04761. arXiv: 2302.04761[cs]. URL: http://arxiv.org/abs/2302.04761 (visited on 07/19/2023) (cit. on p. 14).

[16] Yingqiang Ge, Wenyue Hua, Kai Mei, Jianchao Ji, Juntao Tan, Shuyuan Xu, Zelong Li, and Yongfeng Zhang. *OpenAGI: When LLM Meets Domain Experts*. Issue: arXiv:2304.04370. June 18, 2023. arXiv: 2304.04370[cs]. URL: http://arxiv.org/abs/2304.04370 (visited on 07/28/2023) (cit. on p. 14).

[17] Evan King, Haoxiang Yu, Sangsu Lee, and Christine Julien. *Sasha: creative goal-oriented reasoning in smart homes with large language models*. Issue: arXiv:2305.09802. May 16, 2023. DOI: 10.48550/arXiv.2305.09802. arXiv: 2305.09802[cs]. URL: http://arxiv.org/abs/2305.09802 (visited on 08/14/2023) (cit. on p. 14).

[18] *7 fundamental UX design principles all designers should know - UX Design Institute*. Running Time: 363 Section: Design. June 22, 2022. URL: https://www.uxdesigninstitute.com/blog/ux-design-principles/ (visited on 03/27/2024) (cit. on p. 16).