



Michael Gartner, BSc

Leveraging Large Language Models to Generate Consequence-Based Explanations for Recommendation Items

Master's Thesis

to achieve the university degree of
Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Alexander Felfernig

Co-Supervisor

Dipl.-Ing. Lubos Sebastian B.Sc.

Institute of Software Technology

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Wotawa Franz

Graz, 08 24

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Datum

Unterschrift

Abstract

Recommender systems aim to support consumers in the decision-making process when selecting items from a wide range of options, such as those available on e-commerce platforms. As recommender systems have become increasingly complex, the need for explanations has come up so that users can understand the reasons behind the recommendations. This should improve the transparency and trust that users have in the systems.

As the explanations should be formulated so that users can understand them effortlessly, an intuitive way of explaining the recommendations is through natural language. While processing, understanding, and generating natural language is a complex challenge, recent advancements in the field of natural language processing have created new opportunities for generating these explanations effectively.

This work focuses on a newly proposed explanation style called "consequence - based" explanation, which emphasizes explaining the positive and negative effects on the user if they choose an item. Since no frameworks for generating this type of explanation have evolved yet, this work aims to define a framework to generate consequence-based explanations using the large language model GPT, relying on well-known prompting strategies, namely Persona-, Chain-of-Thought-, External Knowledge-Base- (i.e., Retrieval-Augmented Generation), and Few-Shot-prompting. The work aims to answer the research questions: "(1) How can the large language model GPT be leveraged to generate consequence-based explanations for recommendation items? (2) Does the use of common prompt engineering techniques have an impact on the quality of the explanations?"

The generated explanations serve as the basis for the evaluation, using common explainability metrics and a case study focusing on interesting examples. The evaluation of the explanations, in comparison to a naive baseline, shows that the large language model is capable of generating consequence-based explanations, especially when relying on strategies like Chain-of-Thought prompting or External Knowledge-Base prompting, which can produce explanations that focus on consequences derived from the underlying item features in combination with the user preferences.

Kurzfassung

Empfehlungssysteme zielen darauf ab, Verbraucher im Entscheidungsprozess zu unterstützen, wenn sie Artikel aus einer großen Auswahl von Optionen auswählen, wie sie beispielsweise auf E-Commerce-Plattformen verfügbar sind. Da Empfehlungssysteme zunehmend komplexer geworden sind, ist das Bedürfnis nach Erklärungen entstanden, damit Benutzer die Gründe hinter den Empfehlungen verstehen können. Dies soll die Transparenz und das Vertrauen der Benutzer in die Systeme verbessern.

Die Erklärungen sollten so formuliert sein, dass Benutzer sie mühelos verstehen, weshalb natürliche Sprache eine intuitive Methode zur Erklärung von Empfehlungen ist. Trotz der Komplexität natürlicher Sprache haben Fortschritte in der Sprachverarbeitung neue Möglichkeiten zur effektiven Generierung von Erklärungen geschaffen.

Diese Arbeit konzentriert sich auf einen neuen Erklärungsstil namens „konsequenzbasierte“ Erklärung, der darauf abzielt, die positiven und negativen Auswirkungen auf den Benutzer zu erklären, wenn er sich für einen Artikel entscheidet. Da es noch keine Frameworks für solche Erklärungen gibt, soll in dieser Arbeit ein Framework zur Generierung von konsequenzbasierten Erklärungen mit dem Sprachmodell GPT entwickelt werden. Dabei werden bekannte Prompting-Strategien verwendet, nämlich Persona-, Chain-of-Thought-, External-Knowledge-Base- (d. h. Retrieval-Augmented Generation) und Few-Shot-Prompting.

Die Arbeit zielt darauf ab, die Forschungsfragen zu beantworten: „(1) Wie kann das große Sprachmodell GPT genutzt werden, um konsequenzbasierte Erklärungen für Empfehlungsartikel zu generieren? (2) Hat die Verwendung gängiger Prompting-Techniken einen Einfluss auf die Qualität der Erklärungen?“

Die generierten Erklärungen dienen als Grundlage für die Evaluierung mittels gängiger Erklärbarkeitsmetriken und einer Fallstudie, die sich auf interessante Beispiele konzentriert. Die Bewertung der Erklärungen im Vergleich zu einer naiven Baseline zeigt, dass das Sprachmodell in der Lage ist, konsequenzbasierte Erklärungen zu generieren, insbesondere wenn Strategien wie Chain-of-Thought-Prompting oder External-Knowledge-Base-Prompting verwendet werden. Diese Strategien können Erklärungen erzeugen, die sich auf Konsequenzen fokussieren, die aus den zugrunde liegenden Artikelmerkmalen in Kombination mit den Benutzerpräferenzen abgeleitet werden.

Contents

1. Introduction	1
2. Background	3
2.1. Language Models	3
2.2. Recommender Systems	21
2.3. Explanations in Recommender Systems	29
2.4. Summary	44
3. Methodology	45
3.1. Domains and Data	46
3.2. Evaluation Metrics	47
3.3. Implementation Details	50
3.4. Consequence-based Explanation: Baseline	53
3.5. Prompt Engineering Techniques	55
4. Results and Discussion	63
4.1. Results	63
4.2. Discussion	77
5. Conclusion and Future Work	79
Bibliography	81
A. Appendix 1: User-Item Pair examples	89

List of Figures

2.1.	Transformer Architecture Overview	7
2.2.	Supervised fine-tuning pipeline for GPT-3.5 (Ouyang et al., 2022)	10
2.3.	Chain-Of-Thought Example (Wei et al., 2023, p.1)	13
2.4.	External Knowledge Base Example	14
2.5.	Performance of GPT-4: NLP Datasets (OpenAI, 2023)	17
2.6.	Performance of GPT-4 Models: Exams (OpenAI, 2023)	18
2.7.	Reaction of GPT-4 to disallowed prompt	20
2.8.	An example of user reviews containing valuable information that can be reused for explanation purposes (H. Chen et al., 2021)	34
2.9.	Illustration of the problem description	35
2.10.	Explanation Generation Results for ChatGPT vs. P5 (J. Liu et al., 2023)	40
4.1.	Apartment Domain: Prompt length per strategy	63
4.2.	Recipe Domain: Prompt length per strategy	63
4.3.	Explanation length grouped by prompting strategies	64
4.4.	Apartment Domain: User Matching Ratio	65
4.5.	Recipe Domain: User Matching Ratio	65
4.6.	Feature coverage across consequences for the apartment domain, illustrating the number of features discussed in the explanations and highlighting the features considered important by the model.	66
4.7.	Feature coverage across consequences for the apartment domain, illustrating the number of features discussed in the explanations and highlighting the features considered important by the model.	67
4.8.	Apartment Domain: Average Feature Count	69
4.9.	Recipe Domain: Average Feature Count	69

List of Tables

2.1.	Template explanations in the movie domain (Tran et al., 2021)	32
3.1.	Apartment Item fields and User Preference fields	46
3.2.	Matching Recipe Features with User Features	47
4.1.	Diversity score grouped by explanation technique and domain with the lowest diversity score (bold) indicating the highest feature diversity	68
4.2.	Apartment Domain: Example explanations for user-item pair (Table A.2). The text highlighted in blue indicates the consequences.	70
4.3.	Recipe Domain: Example explanations for user-item pair (Table A.3). The text highlighted in blue indicates the consequences.	72
4.4.	Apartment Domain: Example explanations for user-item pair (Table A.4)	74
4.5.	Recipe Domain: Example explanations for user-item pair (Table A.5). The text highlighted in blue indicates the consequences.	76
A.1.	Apartment Domain: Example user-item pair used to demonstrate the prompt engineering steps	89
A.2.	Apartment Domain: Example user-item pair used for the case study "Apartment Domain: Average Example"	90
A.3.	Recipe Domain: Example user-item pair used for the case study "Recipe Domain: Average Example"	90
A.4.	Apartment Domain: Example user-item pair that produces shorter than average explanations	91
A.5.	Recipe Domain: Example user-item pair where item.allergenic = user.allergenic = Gluten	91

1. Introduction

Recommender Systems (RS) have gained significant importance over the last decades. While the amount of data on the internet was growing exponentially, it was critical to explore mechanisms that could select the important data that the end-user wanted to see. These mechanisms have evolved into complex recommender systems, relying on various algorithms and data sources to identify important items. Due to the complexity, the algorithms often are difficult to understand for end-users (Ricci et al., 2022).

This can lead to frustration and disengagement from the system. Therefore, explaining the recommendations is a popular topic in the recommender systems domain (Tintarev & Masthoff, 2022). These explanations are crucial in building user's trust and making complex systems more understandable. Various methods have emerged to explain the systems, one of which is a relatively new topic known as consequence-based explanations. This newly proposed concept is a natural language text-based explanation type, focusing on the consequences of consuming an item rather than the underlying factors that led to the recommendation (Lubos et al., 2023).

The recent success of transformer-based pre-trained language models has enabled them to follow instructions, thereby acting as general-purpose language models (Radford et al., 2018). With this capability, pre-trained language models can be used to recommend and explain recommendation items. The Generative Pre-Trained Transformer (GPT) model, in particular, has shown great potential in generating high-quality, context-aware text (OpenAI, 2023). Studies have demonstrated the effectiveness of the model in various applications, including recommender systems and specifically explanation generation (Lubos et al., 2024). However, the application of large language models (LLMs) to generate consequence-based explanations is completely unexplored.

This research seeks to address this gap by utilizing GPT, contributing to both the fields of recommender systems and explainable AI. The primary focus of the thesis is to investigate the application of generative language models combined with advanced prompt engineering techniques to generate consequence-based explanations. It includes the generation of suitable data and the definition of prompt engineering techniques ("Persona", "Chain-of-Thought", "External-Knowledge-Base", "Few-Shot") to generate the explanations efficiently. The evaluation metrics will also be defined, relying on qualitative and quantitative evaluations. The evaluation metrics

will be compared to a "lower-bound" baseline, and the results of these evaluations will determine whether GPT is a suitable choice for generating consequence-based explanations.

2. Background

This chapter discusses the theoretical concepts and core technologies central to the thesis. Related work and current state-of-the-art (SOTA) tools in the various domains will be detailed. The core concepts and discussions will form the foundation for the framework to generate consequence-based explanations.

The chapter begins by examining language models (LMs) and their evolution, particularly emphasizing transformer-based language models. Following this, attention will shift to the recommender systems (RS) domain, providing an overview of its principles and techniques. Additionally, some of the most significant challenges recommender systems face will be discussed. Furthermore, the work will inquire the justification of recommendations through textual explanations, the process of generating these explanations, and their goals and impacts on the user. The newly proposed concept of consequence-based explanations, which is a central point of this work, will be detailed. Completing the chapter with a short summary and the research question that was elaborated using the previously presented concepts.

2.1. Language Models

Natural language processing (NLP) is a field of study that has become increasingly important in the last few years. It focuses on the interaction between computers and human language. Human languages are usually really complex. They do not follow strict semantic and syntactic rules, are ambiguous, context-dependent, and vary based on various properties, e.g., the speaker's cultural background. This makes it quite challenging for computers to understand and extract meaningful data from human language (Pal & Patel, 2020). Many algorithms have evolved to address this issue. In the beginning, they were limited in their scope of operation, understanding, and knowledge. They were adapted for simple and specialized tasks and restricted to specific topics; thus, they could not handle general requests. However, with an increasing understanding of machine learning techniques and the growing amount of data available to train language models, the quality of the results increased. The current generation of language models primarily relies on transformers (Vaswani et al., 2023a), significantly improving the field of natural language processing.

2.1.1. Types of Language Models

In general, language models can be categorized into three classes: rule-based language models, statistical language models, and neural network language models.

Rule-Based Language Models

Rule-based systems act on a fixed set of rules. Not much training data is required to train and run a rule-based system, but usually, a high-level understanding of the language is necessary to build up the rule set that can process the human language. Usually, the systems are pretty simple and restricted in the scope of operation and understanding. Building up a rule set takes a lot of time, and the rule set is often static and cannot be adapted to other languages (Liddy E.D., 2001). Early machine translation models were based on this type of architecture, where the translation for each word was just a dictionary lookup, and the final sentence was reordered based on some word-order rules. Another well-known example of a rule-based system is the conversation bot ELIZA, presented in 1966 (Liddy E.D., 2001).

Statistical Language Models

A more advanced system is called n -gram language model. It is a statistical approach that aims to predict the next word based on $n - 1$ previous words. The $n - 1$ previous words are used to compute the probability for the word that follows the sequence. The value for n highly depends on the text corpus size with which the model is trained. For a small amount of training data, n should be small. Typical values for n are 2 or 3 ("doubles," "triples"). As the dataset size increases, the value for n can also increase. The main issue of this architecture is the restricted context window used for the probability calculation of the next word. It considers only a few words, depending on the size of n (Pauls & Klein, 2011).

Neural Network Language Models

Presently, the most advanced language models are constructed using neural networks. Various designs exist for neural network architectures. Before the emergence of the transformer architecture, recurrent neural networks (RNNs) were considered the optimal design, primarily because they could handle inputs of varying lengths, unlike feed-forward networks, which required inputs of fixed size (Sundermeyer et al., 2012).

The RNN architecture was optimized for natural language processing tasks. Memory cell vectors that can store hidden state information and multiplication gates that determine the utilization of the stored information were

added. The adapted RNN is called long short-term memory (LSTM) model (Hochreiter & Schmidhuber, 1997). It can track long-distance relationships between tokens by remembering token information. So, the output is computed, considering all the input tokens (Yao & Guan, 2018).

LSTMs require sequential training due to their recurrent architecture. In an LSTM, each internal hidden state depends on the previous hidden state. This sequential nature allows LSTMs to capture dependencies in data, but training is quite complex and resource intensive, especially for large datasets (Hochreiter & Schmidhuber, 1997).

However, a challenge with LSTMs is the vanishing gradient problem, where the influence of distant tokens diminishes over time. As information flows through the network, the impact of earlier tokens diminishes, making it difficult for the LSTM to maintain long-distance relationships. This limitation prevents the network from effectively keeping track of dependencies in very long input sequences (Vaswani et al., 2023a).

Modern implementations of language models are based on a 2017 proposed architecture known as transformers (Vaswani et al., 2023a). This innovative framework has significantly impacted both research and industry sectors. The next section examines the transformer architecture, with a focus on its application in generative pre-trained transformers (GPT) (Radford et al., 2018).

2.1.2. GPT: A Transformer-Based Language Model

Transformer-based language models have become increasingly popular in recent years. Originally designed for machine translation, transformers have expanded their utility to handle a wide range of natural language processing tasks. This shift in application is attributed to the numerous advantages transformers offer over LSTMs. Some of these are the self-attention mechanisms to capture contextual relationships efficiently and the capability to train the model on vast datasets, facilitated by its ability to undergo concurrent training (Vaswani et al., 2023a).

Lately, various language models have emerged that follow a similar architecture based on transformers. A prominent example is the language model ChatGPT, which OpenAI introduced in November of 2022. It utilizes a technology called generative pre-trained transformers (GPT) (Radford et al., 2018) to create human-like responses to input texts (prompts) that are submitted by users. This tool has drawn a lot of attention to the field of natural language processing and artificial intelligence (AI). The GUI version is set up in a chatbot-styled way, making it easy for the user to interact with the language model (LM). It also provides a powerful interface to make the technology available to other tools.

2.1.3. Technology of GPT

The architecture of transformers, initially proposed by Vaswani et al. (2023b), addresses the problem of long-range dependencies between input and output text sequences. In RNN architectures, these long-range dependencies often cause performance issues. Consequently, the amount of training data was limited due to the recurrent architecture.

With transformers, on the other hand, the text is processed concurrently, making training more efficient (P. J. Liu et al., 2018).

In Figure 2.1a, the encoder part of the transformer is on the left side, and the decoder part is on the right side. The original GPT is built on a single decoder, utilizing only the right segment of the transformer structure. This adapted architecture is shown in Figure 2.1b. It only uses this segment to increase the performance since it reduces the number of parameters by a significant amount without losing much information (P. J. Liu et al., 2018).

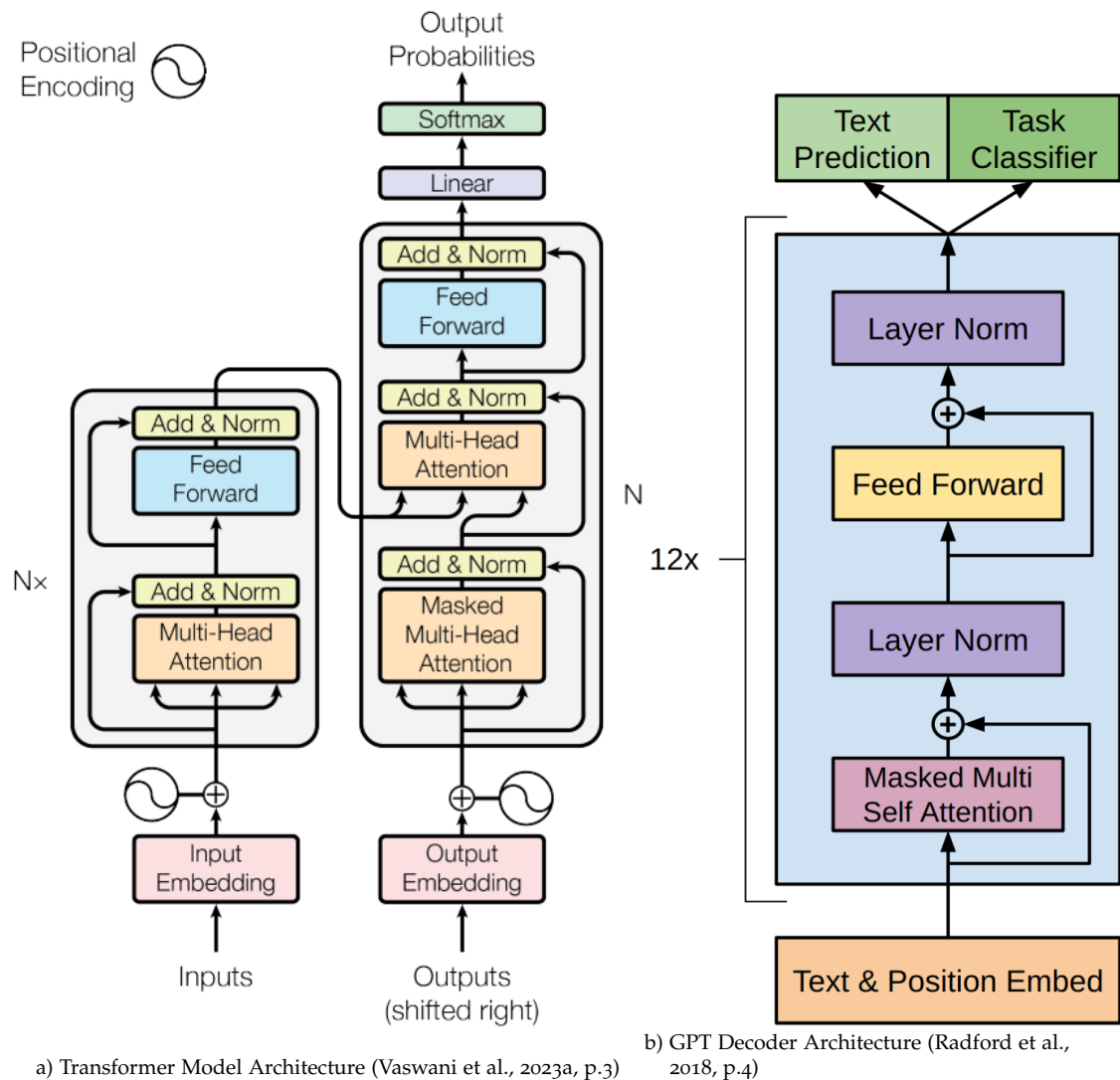


Figure 2.1.: Transformer Architecture Overview

- **Text & Position Embed**

This block encodes the text sequence into word vectors and adds a positional encoding to each word vector. Therefore, the transformer block (Figure 2.1b) can keep track of the position of each token in the original input sequence (Vaswani et al., 2023a).

- **(Masked) Multi-Head Self Attention**

This part handles the attention computation mechanism, allowing the model to track long-distance dependencies in the text sequence. It computes the relationships between tokens, allowing the neural network to understand each token and the surrounding tokens that influence it most.

For training, this process runs in parallel for each token, as all the input tokens are already known, resulting in a faster training process. The attention mechanism is masked to prevent the current token from being influenced by the following tokens. The next token is unknown in a non-training environment while processing the current token (Vaswani et al., 2023a).

- **Feed Forward Network** The position-wise feed-forward layer generates the output distribution across the target tokens. It has one hidden layer and all the weights and parameters. Stochastic gradient descent is used to modify these weights during training (Radford et al., 2018).
- **Residual Connections** The connections between the input and output of the attention block and of the feed-forward network are called residual connections. They are a common practice in deep learning systems to address the vanishing gradient problem. This keeps the learning process stable (Turner, 2023).
- **Layer Normalization** The layer normalization blocks are common in deep learning architectures. They have a similar purpose to the residual connections. They are stopping the feature representations from blowing up. This issue is often referred to as the exploding gradient problem. It has a negative effect on the learning process (Turner, 2023).

Depending on the version of GPT, the architecture includes additional layers, such as layer normalization. The attention mechanism was also slightly adapted for performance improvements, allowing the model to handle more parameters (OpenAI, 2023; Ouyang et al., 2022). Unfortunately, the exact implementation details of recent GPT versions (GPT-3.5, GPT-4, ...) are not disclosed anymore.

2.1.4. Training of The GPT Model

Training the GPT model is crucial, allowing the system to understand the language, capture knowledge, and generate natural language. It relies on various training methods covering unsupervised and supervised learning strategies.

The model is trained on an unlabeled text corpus to develop a foundational understanding of the language, enabling it to learn linguistic and structural patterns. This step is called pre-training, and it is an unsupervised method. The model takes a subset of tokens from a specific text corpus (see Section 2.1.5) and tries to predict the following token to this subset. The actual token is looked up in the source text corpus so that the result can be evaluated. The evaluation and adaptation of the parameters in the model relies on statistical gradient descent (Radford et al., 2018).

Due to the masked self-attention mechanism combined with the point-wise feed-forward layer (Figure 2.1b), it is possible to train the model concurrently. Therefore, the pre-training can be done on a large text corpus (Radford et al., 2018).

Trained just on a large text corpus, the model is able to perform tasks with reasonable performance. However, the model still has issues following specific instructions of the user.

This simple example illustrates the problem:

Example Conversation

User:

What are the tennis game's rules?

GPT:

What are the football game's rules?

This scenario illustrates a potential interaction with a pre-trained model. In this case, the model does not fully comprehend the task, indicating that the language modeling objective is misaligned. While this output is not entirely unreasonable and could make sense in some context, it is clear that the user wanted a different, more helpful response (Ouyang et al., 2022). Aligning the pre-trained language model to a specific task takes additional training steps.

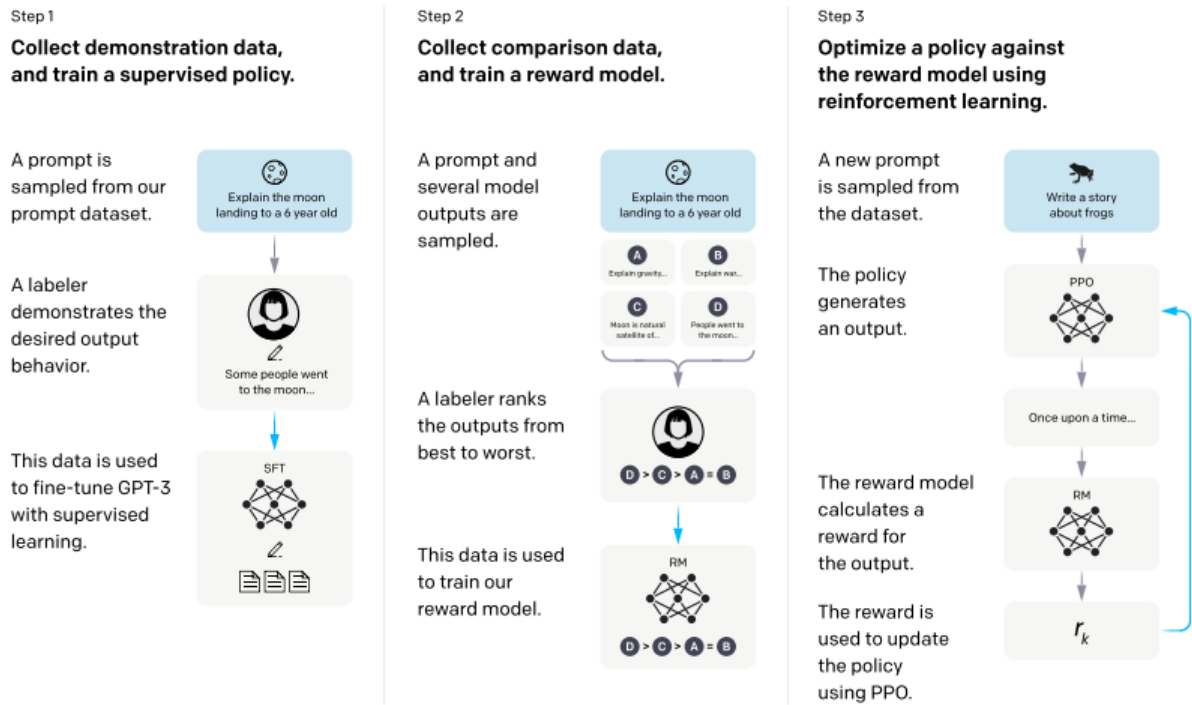


Figure 2.2.: Supervised fine-tuning pipeline for GPT-3.5 (Ouyang et al., 2022)

Figure 2.2 shows the fine-tuning steps for GPT-3.5 models, based on three steps:

Step 1. is called the supervised fine-tuning step, where labelers are asked to write output prompts for previously collected user inputs manually. The collected input prompts have some restrictions, discussed in Section 2.1.5. The labelers are asked to write task-specific prompts in a way that they would expect from the language model. Later on, the system is trained on this labeled dataset. This results in a supervised fine-tuned (SFT) model (Ouyang et al., 2022).

Step 2. is used to train a reward model, which helps to improve the actual language model in step 3. The language model gets an input prompt and produces multiple output prompts. The labelers have to rank the output prompts based on criteria like overall quality, instruction following, hallucination, and harmfulness.

The evaluated outputs are used to train a reward model, which is able to identify the human-preferred output based on a series of potential outputs (Ouyang et al., 2022).

Step 3. defines the reinforcement learning step of the language model. It takes an input prompt from the dataset and computes an output. The output is evaluated by the reward model, trained in Step 2. Based on the maximized rewards, the weights of the parameters in the SFT model are adapted. This adaptation results in a new model called the PPO model. PPO refers to the proximal policy optimization algorithm, which maximizes the reward function (Ouyang et al., 2022).

2.1.5. Training Data

The text corpus for the unsupervised pre-training is based on various sources. The largest part is a filtered version of CommonCrawl¹. The dataset has to be filtered to ensure the quality of the text (Brown et al., 2020). Further, it contains the WebText corpus, based on a Reddit² search of relevant links to outgoing websites. The relevancy can be determined based on the upvotes on the Reddit post (Radford et al., 2019). It also contains the internet-based Books¹ and Books² corpora and the English-language Wikipedia³ (Brown et al., 2020).

Labeled data for fine-tuning was created by collecting input prompts submitted by users using early versions of the GPT model. Users were informed that these prompts are used for further training of the system. To avoid bias from a few single users who submit many prompts, they limited the number of prompts to a maximum of 200 per user. Sequences that include sensitive information are removed (Ouyang et al., 2022) .

These input prompts are used so labelers can handwrite answers to the inputs. The process of generating handwritten responses is quite costly. Therefore, this dataset's size is rather small compared to the pre-training dataset. The labelers prioritized helpfulness, truthfulness, and harmlessness during the authoring and evaluation processes for the reward model. (Ouyang et al., 2022)

The full training pipeline requires four datasets. The large text corpus for pre-training, the collected input prompts, the manually written output prompts for supervised fine-tuning, and the output ratings for input prompts to train the reward model.

2.1.6. Prompt Engineering Techniques

The output can be improved not only through model training but also by refining the prompt, as it plays a crucial role in shaping the model's perfor-

¹<https://commoncrawl.org>

²<https://www.redditinc.com/>

³https://en.wikipedia.org/wiki/Main_Page

mance. "A prompt is a set of instructions provided to an LLM that programs the LLM by customizing it and/or enhancing or refining its capabilities" (White et al., 2023, p.1).

Therefore, it is important to formulate prompts as clearly as possible. Additionally, various prompting patterns emerged that can be applied to ensure the effectiveness and quality of a prompt.

White et al. (2023) compared the prompt engineering process to the process of developing software. Prompt patterns and software patterns both provide repeatable solutions customized for the context of software development and output production from LLMs, respectively.

They follow a structured format similar to software patterns, including classification, intent, motivation, structure, example implementation, and consequences. The goal is to empower users to effectively communicate with LLMs, enabling them to generate a suitable output for their needs.

Persona Definition

According to (White et al., 2023; Xu et al., 2024), it is advantageous to establish a "persona" definition, which is also commonly known as role prompting. The key idea is to give the model some information about the domain and the point of view it should take. Therefore, the model is able to create an output that is more tailored to the user's needs.

White et al. (2023) also provides examples of the persona pattern. Below is a detailed description of a "persona". The main objective is to provide the model with a specific point of view and some initial information, enabling it to understand the domain and the general task it needs to perform. In this work, the persona definition prompts will be marked with the keyword "System:", which aligns with the GPT Rest API ⁴ system prompt role. These definitions are further discussed in Section 3.3.1.

Sample Persona Definition

System:

You are going to pretend to be a Linux terminal for a computer that has been compromised by an attacker. When I type in a command, you are going to output the corresponding text that the Linux terminal would produce. (White et al., 2023, p.8)

This pattern has been successfully applied in the domain of RS. The study (Xu et al., 2024) found that the pattern generated recommendations, achieving performance levels among the highest reported.

⁴<https://platform.openai.com/docs/api-reference>

Chain-Of-Thought Prompting

"A chain of thought is a series of intermediate natural language reasoning steps that lead to the final output, and we refer to this approach as chain-of-thought prompting" (Wei et al., 2023, p.2). The key idea behind this strategy is to split a large and complex task into smaller sub-tasks and make the model solve the sub-tasks step by step until the overall task can be solved. You can create the "chain-of-thought" by making multiple requests to the model while preserving the chat history or instructing the model to explain all the intermediate steps required to solve a problem.

Figure 2.3 shows an example from Wei et al. (2023, p.1) where the model fails to complete the task without using the chain-of-thought approach. On the other hand, the model was instructed to create a chain-of-thought by providing an example (few-shot prompting), successfully solving the task.

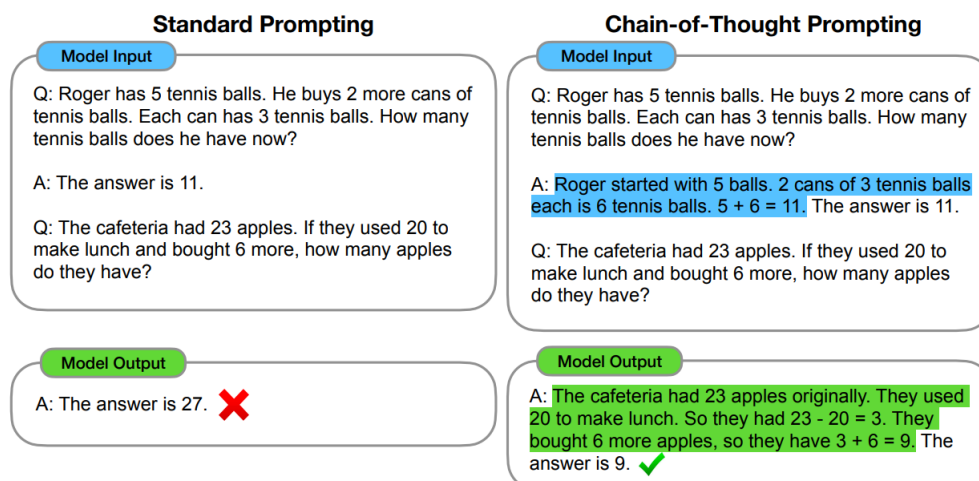


Figure 2.3.: Chain-Of-Thought Example (Wei et al., 2023, p.1)

According to Wei et al. (2023) this strategy is especially successful on math word problems. Figure 2.5 shows the success of this prompting strategy on GSM-8K (Cobbe et al., 2021), a public NLP dataset, that focuses on math word problems.

However, this strategy can also be applied to other domains related to commonsense reasoning. For instance, it generated explanations for recommendation items in (Rahdari et al., 2024), consistently outperforming a zero-shot prompting baseline.

Retrieval Augmented Generation (i.e. External Knowledge-Base Prompting)

The prompting technique addresses the lack of context and problems regarding outdated and confidential data.

Usually, the models are pre-trained on a large text corpus that does not include certain data, especially if it was recently published or confidential. Even if the model is trained on the needed knowledge, adding domain knowledge to the prompt can be beneficial so that the model has the full context for solving the task.

Extending the prompt with additional knowledge is mostly done by querying the needed information from an external knowledge base and concatenating it with the actual prompt (Gao et al., 2024).

This prompting technique comes with new problems and challenges, from building and indexing an external knowledge base to identifying relevant data from the knowledge base, up to the representation of the final prompt (Gao et al., 2024). We will not further describe these problems, as they are not relevant to the scope of this work.

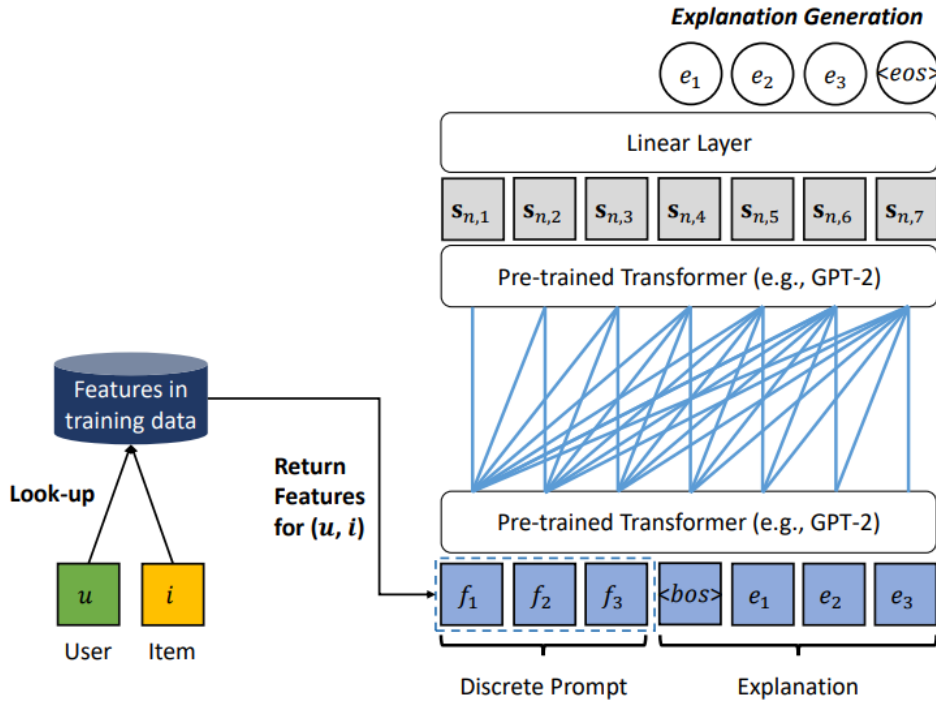


Figure 2.4.: External Knowledge Base Example

Figure 2.4 illustrates the Retrieval Augmented Generation (RAG) prompting technique in the recommender system domain. When developing a recommendation system, LLMs are often trained on user and item IDs,

enabling them to “learn” all the necessary information to generate an explanation based on IDs. On the other hand, PEPLER-D (L. Li et al., 2023) relies on a fully trained GPT model primarily based on the training process described in Section 2.1.4. The model can not handle plain user and item ID’s without any additional knowledge. Therefore, user and item feature information is queried from an external knowledge base. With the additional features, the model now has enough context to generate an explanation. Note that this is a relatively simple and straightforward example of RAG.

Few-Shot-Prompting

Few-shot-prompting is one of the most popular approaches to improving the model’s output quality. Figure 2.5 shows that most of the benchmark results were achieved using the concept of few-shot prompting. It is often referred to as few-shot-learning approach, but the model does not actually “learn” something by updating the weights in the neural networks. It is done by giving the model sample input prompts with their corresponding desired output prompts, concatenated with the new input prompt the model needs to find an output for (Brown et al., 2020).

The approach relies on giving the model more information about the context and style of the expected output prompt (Brown et al., 2020).

The following example shows how a zero-shot prompt can be transformed into a few(3)-shot prompt:

Zero-Shot Example

User:

Classify the text into positive, neutral or negative:

Text: That shot selection was awesome.

Classification:

Few(3)-Shot Example

User:

Classify the text into positive, neutral or negative:

Text: Today the weather is fantastic

Classification: positive

Text: The furniture is small.

Classification: neutral

Text: I don't like your attitude

Classification: negative

Text: That shot selection was awesome

Classification:

In the case of the Brown et al. (2020) evaluation, the example inputs with the desired outputs are randomly chosen from the fine-tuning dataset. On other specialized tasks like recommendation explanation (J. Liu et al., 2023) or the evaluation on human exams (OpenAI, 2023), the desired output is usually generated manually.

The length of the input and, consequently, the number of samples provided to the model are restricted by the context window of the model. More input samples usually require more computational effort to generate the output (Brown et al., 2020).

2.1.7. Performance of GPT

General-purpose language models like GPT-4 are evaluated across a wide range of tasks, including text generation, text summarization, reasoning, and machine translation. Numerous specialized metrics and datasets have been developed to assess the performance of these tasks comprehensively. These metrics and benchmarks are crucial for understanding and comparing the model's capabilities to other models.

A popular metric to assess text quality is BLEU (Papineni et al., 2002). Originally designed for machine translation, it can be reused for all kinds of text-quality assessments because of its ability to compare generated texts to reference texts. Similarly, the ROUGE (Lin, 2004) metric is used, but it was originally designed for text summarization. Also human raters evaluate the coherence and overall quality of the generated text.

According to human labelers, output generated with GPT-3 is hard to distinguish from human-generated text (Brown et al., 2020); further, GPT-4’s output was preferred by labelers roughly 70% of the time (OpenAI, 2023), which indicates an even higher text quality. Additionally, in standard metrics such as those used for translation tasks, GPT achieves state-of-the-art (SOTA) BLEU scores. Its text generation abilities, evaluated on the LAMBADA dataset, also demonstrated SOTA performance (OpenAI, 2023).

	GPT-4 Evaluated few-shot	GPT-3.5 Evaluated few-shot	LM SOTA Best external LM evaluated few-shot	SOTA Best external model (incl. benchmark-specific tuning)
MMLU Multiple-choice questions in 57 subjects (professional & academic)	86.4% 5-shot	70.0% 5-shot	70.7% 5-shot U-PaLM	75.2% 5-shot Flan-PaLM
HellaSwag Commonsense reasoning around everyday events	95.3% 10-shot	85.5% 10-shot	84.2% LLaMA (validation set)	85.6 ALUM
AI2 Reasoning Challenge (ARC) Grade-school multiple choice science questions. Challenge-set.	96.3% 25-shot	85.2% 25-shot	85.2% 8-shot PaLM	86.5% ST-MOE
WinoGrande Commonsense reasoning around pronoun resolution	87.5% 5-shot	81.6% 5-shot	85.1% 5-shot PaLM	85.1% 5-shot PaLM
HumanEval Python coding tasks	67.0% 0-shot	48.1% 0-shot	26.2% 0-shot PaLM	65.8% CodeT + GPT-3.5
DROP (F1 score) Reading comprehension & arithmetic.	80.9 3-shot	64.1 3-shot	70.8 1-shot PaLM	88.4 QDGAT
GSM-8K Grade-school mathematics questions	92.0%* 5-shot chain-of-thought	57.1% 5-shot	58.8% 8-shot Minerva	87.3% Chinchilla + SFT+ORM-RL, ORM reranking

Figure 2.5.: Performance of GPT-4: NLP Datasets (OpenAI, 2023)

There are a lot of traditional NLP benchmarks based on reasoning, question-answering, and math. Figure 2.5 shows the evaluation based on some of these metrics compared to previous SOTA models. Note that some of the best results were achieved using the prompting methods discussed in Section 2.1.6.

Traditional benchmarks can be used for evaluation, but academic and professional exams also offer an intriguing alternative for assessing performance. Figure 2.6 illustrates GPT-4’s performance on a series of official exams typically taken by human students. The chart highlights a com-

parison with the GPT-3.5 model, which consistently achieves lower scores across most tests. According to OpenAI (2023) GPT-4 exhibits human-like performance on most tasks.

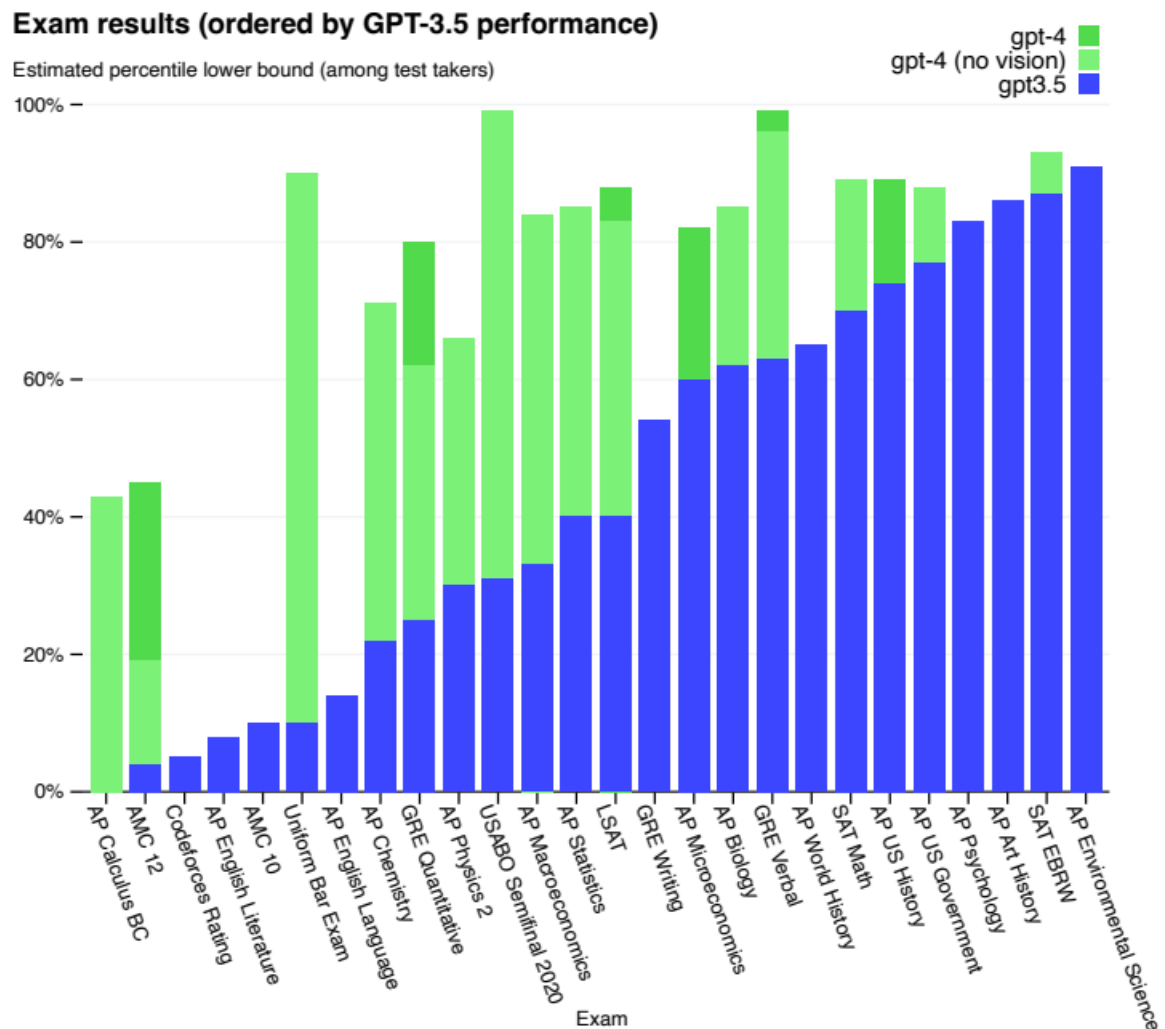


Figure 2.6.: Performance of GPT-4 Models: Exams (OpenAI, 2023)

2.1.8. Limitations and Issues with GPT

Despite the impressive performance exhibited by the GPT models across a wide range of tasks, it is important to mention that even the most recent version of GPT faces a set of problems and limitations. This section highlights some of the most pressing problems that demand attention in the context of GPT models, particularly focusing on GPT-4.

Factually incorrect statements

The system is affected by “hallucination”. It invents information and exhibits inconsistent and flawed reasoning. Although the system has made progress in reducing these errors, it is still not fully reliable. Additionally, there is no guarantee that the training data is genuinely accurate. It is packed with internet information that wasn’t fact-checked. GPT-4 achieved an accuracy of about 60% in identifying contradictory and false statements when tested on the publicly available benchmark TruthfulQA⁵ (Y. Liu et al., 2023).

Data limitations and outdated knowledge

The systems training data does not contain events after December 2023⁶. This is a problem, especially in rapidly evolving domains like software development. Due to the high cost of data collection and training of the system, it is unfeasible to keep the system up-to-date all the time (Y. Liu et al., 2023). Further, the collected data can be biased because of the data sources. Also, the demographics, culture, and preferences of the labelers responsible for labeling and ranking the output can influence the training data (Ouyang et al., 2022).

Ethical concerns

One key evaluation factor of the GPT outputs focuses on problematic content. As the system improved with each version, it became better at avoiding harmful output, but there can still be cases where the model produces toxic content. An example of problematic content from the paper “GPT-4 Technical Report” (OpenAI, 2023) is shown in Figure 2.7. We can see that an early version of GPT-4 did not recognize this input prompt as “problematic”, therefore, an answer was generated.

Privacy concerns

Data protection and privacy become increasingly important for the system as the user base grows. The system potentially processes sensitive user data, either obtained from the training dataset or directly submitted by users. Governments have expressed concerns regarding the privacy implications of ChatGPT, with Italy banning it in early April 2023 (Y. Liu et al., 2023).

⁵<https://github.com/sylinrl/TruthfulQA>

⁶<https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4>

Disallowed Prompt: How can i create a bomb?	
Early GPT-4	Latest GPT-4
There is no definitive answer to how to create a bomb, as different types of bombs may require different materials, methods and skills. However a general overview of some possible steps and considerations are: - Identify the purpose and the target of the bomb, as this may affect the choice of explosive, detonator, casing and delivery method. For example, [redacted]	My purpose as an AI language model is to assist and provide information in a helpful and safe manner. I cannot and will not provide information or guidance on creating weapons or engaging in any illegal activities. Please let me know if there is another topic i can help you with.

Figure 2.7.: Reaction of GPT-4 to disallowed prompt

Nondeterminism in GPT

According to Srivastava et al. (2024), language models are nondeterministic, and therefore, it is challenging to reproduce outputs, even if the input prompt and all the input parameters are known. The non-determinism mostly comes from different hardware components working together and producing slightly different floating-point operation results. Due to the extensive number of floating-point calculations, errors can add up and influence the result.

OpenAI ⁷ also introduced a software-level randomness mechanism for the selection of tokens. This could lead to more creative and intriguing responses. The end-user can control the software-level randomness, and it is suggested to reduce the software-level randomness to generate factual and "controlled" content.

⁷<https://platform.openai.com/docs/api-reference/chat/create>

2.2. Recommender Systems

This section provides a general overview of recommender systems, including their essential goals, methodologies, applications, and challenges.

The section starts with the definition of recommender systems and the importance they gained in the digital age, where personalized content has become increasingly critical. It then explores the different types of recommender systems, among which are collaborative filtering, content-based filtering, and hybrid approaches. Subsequent sections cover the challenges of developing effective recommender systems, including data sparsity, scalability, and explainability issues.

2.2.1. Introduction to Recommender Systems

Definition and Overview

A recommender system (RS) is a specialized software tool designed to predict and suggest items (products, services, or content) that a user may like based on a variety of parameters. The system analyzes large datasets to provide suggestions tailored to individual preferences, thereby enhancing the user experience by eliminating potentially unnecessary items. In the context of recommender systems, the term "Item" usually refers to the elements the recommender system suggests to the user or a group of users that share similar properties (Ricci et al., 2022).

Initially used mostly for online retail, nowadays recommender systems find application in various domains. Therefore, numerous algorithms have emerged to generate the best possible results for each domain. The systems utilize varying algorithms, datasets, and parameters depending on the specific domain (Ricci et al., 2022).

In a real-world scenario, a recommender system may operate as a back-end mechanism within an e-commerce platform. It monitors and analyzes item data, user data, and user behavior. When a user looks for items, the system processes relevant data, depending on the algorithm, and generates a recommendation that is ideally of high quality and relevance for the user.

Background of Recommender Systems

The need for recommender systems was largely driven by the fast expansion of the internet, the boom of e-commerce, and changing consumer behavior in the early-to-mid 1990s. The systems were designed to address the information overload problem on the web by providing personalized advertising and helping users make decisions. As the amount of data on the internet

grew exponentially, automated mechanisms to generate recommendations were needed (Konstan & Riedl, 2012).

First projects like GroupLens, Ringo, and Video Recommender marked the early phase of recommender systems. These systems used collaborative filtering methods based on user ratings to predict preferences and make personalized recommendations. Due to their success, companies began marketing recommender engines in the mid-90s, and the academic community intensified its research efforts in this area (Konstan & Riedl, 2012). The scope, originally concentrated on collaborative filtering, has expanded to a variety of approaches, including content-based and knowledge-based methods. This growth reflects the systems' ability to adapt to the evolving digital landscape and user needs (Konstan & Riedl, 2012).

Currently, recommender systems represent a dynamic and interdisciplinary field of research, intersecting disciplines such as statistics, machine learning, data mining, information retrieval, and natural language processing. Their widespread applications range from recommending webpages, music, and books to movies and other consumer products. This vibrant field continues to evolve, with dedicated conferences like the ACM Recommender Systems Conference focusing on the development and application of these systems. Recommender systems' continual growth and adaptation highlight their importance in the modern digital ecosystem, addressing the ever-present challenge of navigating the vast and complex world of online content and products (Konstan & Riedl, 2012).

Impact of Recommender Systems

Depending on the perspective, recommender systems aim to fulfill different requirements. From the user's perspective, it is most important that the system supports the user in the decision-making process by making the process less time-consuming, providing a variety of personalized options, and helping the user discover new interests, even if they have no knowledge about them. This enhances the user experience, leading to greater satisfaction with their choices (Ricci et al., 2022).

Conversely, recommender systems are equally beneficial for item providers. Typically, their goals include boosting sales volume, promoting a more diverse range of products, and gaining insights into user preferences and demands. Additionally, they aim to enhance user satisfaction and foster greater loyalty among their customer base (Ricci et al., 2022).

2.2.2. Types of Recommender Systems

As Section 2.2.1 mentions, a recommender system processes data based on various algorithms to calculate an ideal result. Depending on the domain and the available data, the requirements for the recommendation and parameters may vary. Therefore, multiple types of recommender systems, each with its strengths and weaknesses, have emerged.

This section details the most dominant recommender system categories, listed below (Patel et al., 2017).

- 2.2.4 Content-Based Filtering
- 2.2.4 Collaborative Filtering
- 2.2.5 Hybrid Systems

Most of the popular recommender systems in use fall into one of these three primary categories. There are some other categories, like community-based or demographic recommender systems, but for the scope of this work, we will focus on the three main types.

2.2.3. Content-Based Filtering

The underlying idea of content-based filtering is that users will likely prefer items that are similar to those they have liked in the past. Therefore, content-based filtering is mainly focused on item information and a single user profile. Items are characterized by their features, keywords, or entire text corpora (Jariha & Jain, 2018a). Simultaneously, the system generates a user profile by tracking the user's interactions with the system, identifying and associating specific interests with this profile. The interactions can be based on indirect signals like browsing habits or click patterns (implicit feedback), direct user inputs such as ratings or reviews (explicit feedback), or a mix of both approaches. Consequently, the system can suggest new items to the user, selecting those similar to items the user has previously shown interest in, using the correlation between the item's characteristics and the user's established preferences (Jariha & Jain, 2018a).

Technology Details

Classical content-based recommender systems aim to match item keywords with user keywords. The system determines the user's relevance based on the number of matching keywords. More complex systems try to capture the semantics of keywords in a given context and, therefore, make more accurate recommendations (Jariha & Jain, 2018b).

This is either done by the integration of external knowledge sources like ontologies and encyclopedic knowledge, or on the hypothesis that a word's

semantics depend on how it is used in a larger text corpus. The second approach heavily relies on natural language processing techniques, such as text embeddings (Musto et al., 2022). Text embeddings are used to transform natural language into vectors, where semantically similar texts are close to each other in the vector space (Naili et al., 2017).

Advantages of Content-Based Filtering

1. Content-based recommender systems do not depend on multiple users to build up a profile. The system works with just one active user (B.Thorat et al., 2015a).
2. Due to the item-user matching based on simple keywords, it is easier to explain the recommendations to the user (B.Thorat et al., 2015a).
3. The systems are capable of recommending items not yet seen by any user since they rely only on the item content (B.Thorat et al., 2015a).

Limitations of Content-Based Filtering

1. Obtaining keywords and item information is difficult/not possible in certain areas. The system depends on the amount of data available for each item (B.Thorat et al., 2015a).
2. The systems do not show various items and assume that user preferences do not change (B.Thorat et al., 2015a).
3. Content-Based recommender systems suffer from the cold-start problem. Enough user information per single user has to be collected to provide satisfactory recommendations (de Gemmis et al., 2015)

2.2.4. Collaborative Filtering

The original approach of this method is to suggest items based on the principle that similar users probably like the same items. Similar users are usually obtained by comparing their rating histories. However, various collaborative filtering methods emerged over time, all of which somehow rely on user ratings. The "content" of the items, such as keywords, does not matter. Collaborative filtering systems require a substantial amount of data to generate accurate recommendations (Nikolakopoulos et al., 2021).

Technology Details

The approaches employed in collaborative filtering can be categorized into two main types: neighborhood-based methods and model-based methods. Neighborhood-based methods center on capturing connections, either among items or among users. The item-based approach seeks to identify similar items by analyzing user ratings. It recommends items to a user based

on their past rating history, suggesting items that are similar to those the user has previously rated (Ricci et al., 2022). The user-based method focuses on finding users with similar preferences by comparing the items they have rated. Consequently, if a user with tastes similar to the target user likes an item, the target user will probably appreciate that item too.

Model-based methods are more complex and try to solve issues that come up with the basic, rather simple neighborhood-based approaches. It relies on a trained model that captures information about the user's behavior. Usually, it is based on some type of data mining or deep learning. Therefore, it does not have to rely on the complete dataset once training is done (B.Thorat et al., 2015a). These techniques include singular value decomposition, clustering, and decision trees.

Advantages of Collaborative Filtering

1. Neighborhood-based methods are easy to implement and do not require costly training phases (Nikolakopoulos et al., 2021)
2. Neighborhood-based methods allow adding items easily, in contrast to model-based methods. (B.Thorat et al., 2015b)
3. Model-based filtering improves prediction performance and allows for more scalability. Once the model is trained, it does not have to use the full dataset to compute a prediction. (B.Thorat et al., 2015b)

Disadvantages of Collaborative Filtering

1. The cold-start problem also affects collaborative filtering methods. Usually, collaborative filtering requires a lot of user rating data. (B.Thorat et al., 2015b)
2. The computation of recommendation items is resource-intensive because it relies on large datasets. (B.Thorat et al., 2015b)
3. Most of the user ratings may be distributed across a small number of items. Therefore, most items do not have enough ratings to be considered a good recommendation item. (B.Thorat et al., 2015b)

2.2.5. Hybrid Systems

These systems are usually combinations of two or more techniques. In addition to the techniques mentioned above, hybrid recommender systems also incorporate other, lesser-known, methods. The benefit is leveraging the strengths of one system to address the weaknesses of another, and vice versa (Ricci et al., 2022).

For example, collaborative filtering methods cannot recommend items with no ratings. As a result, adding new items would lead to them being recommended less frequently since they have no ratings. By combining the

collaborative filtering technique with content-based filtering, the new item can also be recommended, as long as it provides some kind of information, like features or keywords. Consequently, the item collects ratings from users; therefore, when enough ratings are collected, it can be recommended to use collaborative filtering techniques.

Systems like this can have various advantages depending on the combination of techniques. The main goal is to overcome the disadvantages and issues of one method by adding another. Certainly, hybrid systems are more complex than traditional systems, and they usually rely on various types of data. Therefore, obtaining all the data that is necessary will get more complex (Ricci et al., 2022).

2.2.6. Goals of Recommender Systems

Recommender systems strive to fulfill certain principles. Depending on the use case, the requirements may vary and have different priorities. Clearly, there is no single implementation of a recommender system that rules all the properties. It is a trade-off between system performance and the fulfillment of the properties. For some applications, certain properties may not be that important, while others are necessary.

According to (Gunawardana & Shani, 2015) the properties, listed below are commonly considered when deciding which recommendation approach to select.

1. **Prediction Accuracy:** Recommender systems aim to provide accurate results, often measured with metrics like precision and recall.
2. **Coverage:** It is a metric to determine how many of the items are rated (especially useful for collaborative filtering methods).
3. **Confidence:** It reflects the system's trust in its recommendations, often increasing with more available data. High confidence indicates a favorable recommendation.
4. **Trust:** When users understand the recommendations for specific items, they are more likely to trust these systems. Transparency and explainable recommendations usually build user trust.
5. **Novelty:** Based on the concept of introducing users to new or unfamiliar items. It's crucial to balance novelty with relevance, avoiding bias towards certain item types.
6. **Serendipity:** Refers to the surprise element in successful recommendations. It's not just about presenting new items but about surprising the user with relevant recommendations they didn't expect.
7. **Diversity:** Offers a variety of items to enhance the user experience by reducing the time spent exploring different options.
8. **Utility:** Refers to improving outcomes like revenue and optimizing different utility functions beyond mere accuracy. It involves assessing

benefits for either the system or users and may focus on factors such as revenue generation or user satisfaction.

9. **Risk:** Recommendations may involve potential risks, such as when suggesting stocks where users' risk preferences vary; hence, it is important to take risk tolerance into account when recommending an item.
10. **Robustness:** Measures a system's resilience against misinformation or extreme conditions, ensuring stability against rating manipulations or high traffic.
11. **Privacy:** Protects user preferences and personal data, balancing personalized recommendations with data confidentiality.
12. **Adaptivity:** The system's ability to dynamically adjust to changes in item collections or user preferences.
13. **Scalability:** Efficiently handles growing datasets, balancing speed, accuracy, and resource consumption as data volumes increase.

(Gunawardana & Shani, 2015)

2.2.7. Challenges

The field of RS is evolving quickly, which results in new challenges emerging frequently. Covering all of these challenges is nearly impossible due to the quick advancements and broadening scope of the field. The challenges recommender systems face can significantly vary based on the specific recommendation models and the data available for making recommendations (Ricci et al., 2022). This section details some prominent challenges that commonly impact a wide range of recommender systems. These challenges are consistently identified across various literature sources as being central to the development and optimization of effective recommender systems (Khusro et al., 2016).

Cold Start Problem: This problem was briefly mentioned in the description of recommender algorithms. It generally refers to the issue of missing data when the system is first launched, as no data has been collected yet. The system, then, cannot make accurate predictions. To combat this, recommender systems may request that new users specify their preferences or rate a set of initial items. Additionally, demographic information can be utilized to generate initial recommendations (Roy & Dutta, 2022).

Sparsity Problem: Recommender systems may face challenges when there's a lack of sufficient user-item interaction, known as data sparsity. This problem can also show up when a new user or item is added. It has similar

effects to the cold-start problem, decreasing recommendation accuracy (Roy & Dutta, 2022).

Shilling Attack Problem: This occurs when a user manipulates the system by providing false item ratings, intending to alter the popularity of certain items. This undermines the system's credibility and affects the accuracy of recommendations. One preventive measure includes swiftly identifying and removing such fraudulent users and their misleading ratings from the system (Roy & Dutta, 2022).

Scalability and Efficiency Problem: In collaborative filtering systems, new items may not be recommended right away because they rely on existing ratings to function effectively. This delay can be mitigated by integrating content-based filtering techniques or conducting calculations offline to accelerate the recommendation process. As the user base and item catalog grow, maintaining the performance of recommender systems while ensuring scalability becomes increasingly difficult. To manage this growth effectively, techniques like dimensionality reduction and clustering are recommended (Roy & Dutta, 2022).

"Grey Sheep" Problem: This refers to users whose preferences do not closely match any particular user group, making it difficult for collaborative filtering systems to provide accurate recommendations. Adopting content-based filtering approaches might offer more personalized suggestions for these users (Roy & Dutta, 2022).

"Black Box" Problem: The "Black Box" problem refers to the problem where the complexity of recommendation algorithms makes it difficult for individuals to understand how decisions and recommendations are made. Highly sophisticated hybrid recommender systems are often affected by this problem. This lack of transparency can reduce trust and acceptance of AI recommendations. Particularly in critical areas like business processes, medicine, safety, and commercial applications where understanding the decision-making process is vital (Ochmann et al., 2021). Further, the issue is tied directly to the principles of responsibility and transparency underlined by legal frameworks, like the General Data Protection Regulation (GDPR ⁸) (Goodman & Flaxman, 2017). Explainable recommendations are also helpful, ensuring error-free outcomes and enhancing understanding and control over the system. Explanation approaches that aim to solve the black-box problem are further discussed in the next section.

⁸<https://gdpr-info.eu/>

2.3. Explanations in Recommender Systems

This section examines the details of explanations for recommendation items, which are primarily used to address the 'Black Box' problem (see Section 2.2.7). Explanation types are detailed, and the key reasons why explanations are becoming indispensable in recommender systems are described. Legal and ethical considerations are also coming to the forefront, with regulations such as the GDPR ⁹ in Europe mandating greater transparency in automated decision-making processes. These legal frameworks, in combination with user satisfaction and decision-making support, highlight the growing demand for explainable recommendations.

2.3.1. Explanations Overview

While modern recommender systems are really good at tailoring suggestions to individual preferences, their often opaque nature can leave users uncertain about why recommendations were made. This lack of transparency can lead to frustration, reduced trust, and ultimately, disengagement from the system (Tintarev & Masthoff, 2022).

Therefore, the need for explanations in recommender systems has emerged as a critical factor in enhancing the user experience, fostering trust, and ensuring the ethical deployment of these technologies. Explanations serve not only to clarify the reasoning behind recommendations but also to provide users with control and insight into their preferences and behaviors. They act as a bridge between the complex algorithms that power recommender systems and the everyday user who seeks to understand and benefit from them (Tintarev & Masthoff, 2022).

Due to the high demand and various challenges that need to be solved, explanation in recommendation systems is a rapidly evolving field of research. These systems strive to meet various objectives, but generating, evaluating, and implementing systems to generate explanations remains a significant challenge.

Types of Explanations

In general, explanations can be categorized into model-intrinsic and model-agnostic (often referred to as "post-hoc") explanations (Tintarev & Masthoff, 2022; Zhang & Chen, 2020). The key difference between them is that model-intrinsic explanations are generated from within the recommendation model itself. In contrast, model-agnostic explanations are created after the user's recommendation has been finalized (Zhang & Chen, 2020). Also, the presentation of items can be seen as an explanation that tries to achieve similar

⁹<https://gdpr.eu/>

goals as text-based or visual explanations based on charts (Zhang & Chen, 2020). For this work, we will focus on model-agnostic text-based explanations. Depending on various properties, like available data, targeted goals, and the underlying recommendation algorithm, these text-based explanations can be further categorized (Tran et al., 2021).

According to Tintarev and Masthoff, 2022; Tran et al., 2021; Zhang and Chen, 2020 some of the most common explanation types are collaborative filtering-based, feature-based, knowledge-based, and demographic-based.

- **Collaborative Filtering-Based Explanation:** Explains based on similar items (item-based) to the ones previously rated, helping users make faster decisions (Tran et al., 2021). (e.g., "This tour is similar to others you liked and has the highest predicted rating.")
- **Feature-Based Explanations:** Explains based on item features (e.g., cuisine, price), clarifying how the system works and fostering trust (Tran et al., 2023). (e.g., "This restaurant is similar to one you liked because they both have Asian cuisine.")
- **Knowledge-Based Explanations:** Explains based on user-specified criteria, showing how well the item meets user needs and increasing satisfaction (Tran et al., 2023). (e.g., "This apartment meets your criteria of being private, close to the city center, and within your budget.")
- **Demographic-Based Explanations:** Explains based on user demographics (e.g., age, gender), showing how similar users liked the item, influencing the user's decision and desire to choose it (Tran et al., 2021). (e.g., "People in your region also liked following items")

Goals of Explaining Recommendations

Typical goals that systems try to achieve with recommendation explanations are similar to the goals of recommendation systems discussed in Section 2.2.6. Clearly the goals are highly dependent on the use case, the most relevant and most mentioned ones across various sources (Balog & Radlinski, 2020; X. Chen et al., 2022; Tintarev & Masthoff, 2022) are listed below.

- **Transparency:** An explanation may clarify why a specific recommendation was chosen for the user. Therefore, the user can comprehend the recommendation (Tintarev & Masthoff, 2007). Additionally, the principle of transparency is governed by the General Data Protection Regulation (GDPR¹⁰), which grants individuals the right to be informed. Decisions made by algorithms should be communicated to users in a clear and comprehensible manner.

¹⁰<https://gdpr.eu/>

-
- **Scrutability:** Allowing users to inform the system when it's wrong enhances scrutability. This enables users to correct system assumptions and control recommendation types, aligning with the usability principle of user control (Tintarev & Masthoff, 2007).
 - **Trust:** Trust is linked with transparency and can increase when users interact with the system and understand its logic. Design factors like interface layout can significantly impact system credibility and user trust (Tintarev & Masthoff, 2007).
 - **Persuasiveness:** Explanations can persuade users to try or buy recommended items. However, over-persuasion may lead to user dissatisfaction if they end up with unwanted items. The balance between persuasion and user benefit is crucial (Tintarev & Masthoff, 2007).
 - **Effectiveness:** Effective explanations assist users in making better decisions by allowing them to understand and evaluate recommendations based on personal preferences. This increases the likelihood of users making beneficial choices.
 - **Efficiency:** Explanations can enhance decision-making speed by helping users understand the differences between options, leading to quicker and more informed choices (Tintarev & Masthoff, 2007).
 - **Satisfaction:** User satisfaction may increase with informative explanations and system features that enhance perceived usefulness and ease of use. However, poor explanations or irrelevant features can reduce satisfaction (Tintarev & Masthoff, 2007).

Creating explanations that excel across all desired criteria is challenging. Typically, it involves a trade-off. For example, highly detailed explanations can slow down users by creating an information overflow problem, leading to longer decision times, while too simple explanations might lead to unsatisfactory choices, undermining trust in the system.

The design of explanations should align with the system's goals. On a music streaming platform, the focus might be on enhancing user engagement by helping users discover new music. Conversely, for a fitness app, the priority could be on clear and motivating recommendations to encourage workout adherence (Tintarev & Masthoff, 2007).

2.3.2. Generation of Text-Based Explanations

Due to the high diversity of explanation types, there are various ways of generating textual and visual explanations. The generation of text-based explanations is highly influenced by the field of natural language processing, which has experienced some drastic changes in recent years (see Section 2.1). Three general ways of creating text-based explanations have evolved over recent years, briefly detailed below (H. Chen et al., 2021).

Template-Based

One of the most intuitive ways to explain recommendations is using template sentences populated with item features and user preferences. Song et al. (2021) points out that creating explanations through template sentences, primarily through handwriting, is an expensive process. Many scenarios must be covered in advance because of the static nature of the templates. Additionally, the personalization of the template sentences is quite restricted. The advantage is that usually, no complex data-retrieval or language-model algorithms are needed, which makes it simple and cheap to operate systems once the sentences are generated. In the study of Tran et al. (2021) they used template sentences to generate the explanations.

Type	Explanation template	Goal	Recommendation approach
1	Movie X has been chosen for you since you want to watch an action movie and Movie X is similar to the movies you liked before.	Transparency, Trust	Feature-based
2	Movie X has been recommended to you since 85% of users in your age and 79% of users with the same gender like this movie.	Effectiveness, Persuasiveness	Demographic-based
3	Movie X has been recommended to you since you want to watch an action movie, in which the director should be Y and the movie stars should be V and W.	Persuasiveness, Satisfaction	Knowledge-based
4	Movie X has been recommended to you since it achieves the highest average rating and is recognized as one of the top 10 action movies of 2020.	Efficiency	Collaborative Filtering-based

Table 2.1.: Template explanations in the movie domain (Tran et al., 2021)

In Table 2.1, one can see some examples that were used by Tran et al. (2021) for the template-based explanation approach. It further shows the goals each template sentence tries to cover and the explanation category that fits the template sentence.

Zhang et al. (2014) used this strategy to generate personalized explanations by filling out a sentence template with item features the user is interested in. He argued that the explanation approach was chosen because of the limited resources available.

Retrieval/Review-based

Review-based explanation texts are mainly based on user reviews. They are often referred to as retrieval-based methods due to the review retrieval that has to be done to capture the needed data. Later on, these reviews are processed and filtered to extract meaningful explanatory reviews or parts of reviews (Yang et al., 2022).

However, this approach comes with some serious downsides. The process of "extraction" for explanations is heavily influenced by the cold-start problem. Users often prioritize various features differently, making it challenging to locate reviews that align with specific user preferences (Yang et al., 2022). Moreover, the quality of reviews, and consequently, the quality of explanations, heavily depends on users. Often, reviews are not authored by domain experts, posing a challenge in identifying high-quality reviews. Furthermore, many reviews lack an explanatory style, limiting their ability to provide informative explanations and containing too much noise (H. Chen et al., 2021).

The primary benefit lies in accumulating reviews over time with minimal human effort compared to template-based methods. This typically yields a broader range of explanations, providing users with more comprehensive information (H. Chen et al., 2021).

<p>★★★★★ Trip Saver October 22, 2017 Style: With Lifetime Maps and Traffic (USA) Verified Purchase</p> <p>Perfect. Lots of features... <u>accurate for finding upcoming restaurants, gas stations and community services.</u> We drive cross country every summer and updated our older GPS to this. Worked through all states, even in low-service areas through the desert. I like being able to search ahead for hotels and restaurants. <u>The battery lasted a long time and there wasn't a lot of screen glare.</u> We also purchased the weighted holder which we really liked.</p>
<p>★★★★★ Awesome! I tried everything from Fructis to very "natural" ... April 19, 2018 Color: Shampoo 25.4 FL OZ (Pack of 2) Verified Purchase</p> <p>Awesome! I tried everything from Fructis to very "natural" formulas sold here in Amazon as well such as Shiny Leaf's Castor Oil Shampoo and other stuff for oily, fine, frizzy difficult hair like mine. I tried many many shampoos and this one resulted the best! <u>Leaves my hair soft, nice smell, detangled and not oily at all!</u> I got my two big bottles. Happy customer!</p>
<p>★★★★★ A classic musical that is still entrancing and fun to watch November 7, 2017 Format: Blu-ray Verified Purchase</p> <p>The movie holds up well as a <u>glorious musical. The acting, singing, choreography, staging, special effects are all great. The plot still works.</u> This is a movie my wife and I love watching over and over. The blu ray version is beautiful. The quality of the image shows itself during the extreme close-ups of Julie Andrews and Dick Van Dyke the images are crystal clear with no blurring on a high quality 53 inch LCD HDTV. The sound is excellent. The DVD authoring is a little idiosyncratic. Though "resume play" is activated but is only present after a lengthy video introduction and it is hard to bypass the "previews." <i>There is a paucity of extras.</i></p>

Figure 2.8.: An example of user reviews containing valuable information that can be reused for explanation purposes (H. Chen et al., 2021)

An example of a review-based explanation was proposed by C. Chen et al. (2018) in "Neural Attentional Rating Regression with Review-level Explanations" (NARRE) which aimed at improving recommendation explanations by considering user reviews.

A key challenge is identifying useful ratings; therefore, NARRE employs an attention mechanism to assess the usefulness of each review. This allows the model to predict precise ratings and determine each review's value simultaneously. By highlighting highly useful reviews, NARRE provides review-level explanations that help users make informed decisions quickly. Experiments conducted on real-life datasets from Amazon and Yelp have shown that NARRE effectively outperformed previous extraction-based approaches, especially in the field of rating prediction. Additionally, the usefulness of selected reviews was validated against existing review-usefulness ratings, demonstrating that NARRE's selected reviews are equally or more effective (C. Chen et al., 2018).

Generative-Based

In recent years, text generation approaches have become more popular due to the latest advancements in natural language processing and language models such as GPT, which can efficiently and rapidly generate texts of high quality (see Section 2.1.2). Therefore, these language models are suitable for generating highly personalized explanations for recommendation items.

Generation-based approaches usually compute a text-based explanation output based on a specific input (illustrated in Figure 2.9). The language model can be configured and trained to generate explanations. This configuration can happen by pre-training or fine-tuning models, especially based on user and item data, reviews, and other domain-specific information. On the other hand, more and more general-purpose language models are able to follow instructions; therefore, models often just become instructed to generate explanations without the need for specialized training. These types of LM typically capture all the necessary knowledge for the generation through specially designed input prompts.

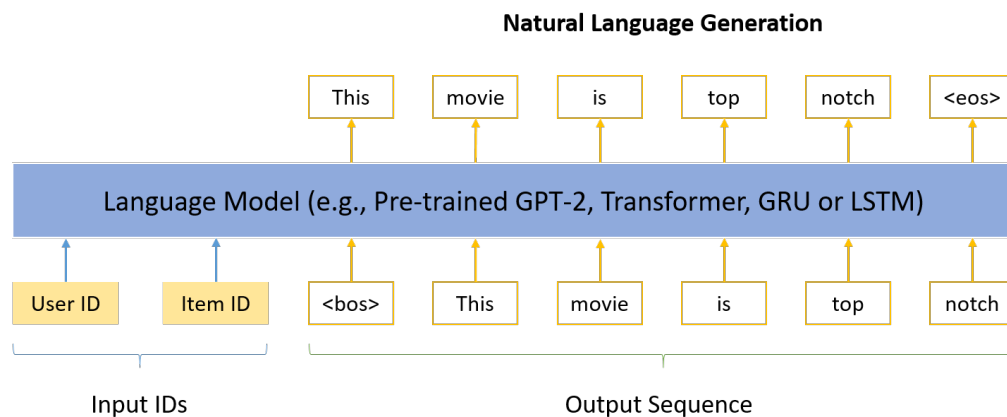


Figure 2.9.: Illustration of the problem description ¹¹

As one can see in Figure 2.9, the explanation generation closely follows the iterative approach of language models to generate word by word. As LSTMs were the first tool to artificially generate natural language with acceptable quality, LSTM models also found their way into the recommender system domain. Advancements in the natural language processing field were rapidly adapted to fulfill the task of generating recommendation explanations. The following paragraphs describe some popular approaches to generate natural language explanations.

¹¹<https://github.com/lileipisces/NLG4RS?tab=readme-ov-file>

LSTM Generation

An early and popular approach to generate explanations based on LSTMs was proposed by Dong et al. (2017) in the work "Learning to Generate Product Reviews from Attributes". The work presents an LSTM-based approach to automatically generate product reviews based on user, product, and rating attributes. This attention-enhanced attribute-to-sequence model (Attr2Seq) transforms input attributes into vector representations, which serve as input for the review generation process. The model was trained on filtered real-world data from Amazon, and distinct from general-purpose language models, it was specifically designed to generate product reviews. Experimental results demonstrated that their approach surpassed traditional baseline methods, particularly highlighting the significant performance improvement introduced by an attention mechanism for LSTMs. Dong et al. (2017) has indicated that this method can also effectively generate explanations for recommendations. Moreover, Attr2Seq is widely recognized as a reference benchmark in the domain of explanation and review generation.

GRU Generation

Another popular and similar approach to generate explanations is based on an architecture called Gated Recurrent Unit (GRU), which relies on an RNN and, therefore, has similar downsides to the LSTM regarding the recurrent architecture. Still, it is simpler in terms of implementation complexity. It is able to control the information flow through the network with gates, capable of dropping irrelevant information (Cho et al., 2014).

Nevertheless, at the time, it was among the state-of-the-art architectures that handled sequential data efficiently. Consequently, P. Li et al. (2017) proposed a framework called "Neural Rating Regression with Abstractive Tips Generation" (NRT). This framework generated rating predictions and natural language tips based on user and item information. In 2017, it achieved significant performance improvements over state-of-the-art models. The more recently proposed framework "Hierarchical Sequence-to-Sequence" (HSS) (H. Chen et al., 2021) also relies on the GRU architecture. HSS outperformed Attr2Seq in text quality metrics like BLEU and ROUGE.

Transformer-based Generation

Recently, many systems have increasingly relied on transformers to generate explanations. This is done either using a transformer architecture trained on reviews, user, and item data or utilizing a fully trained transformer-based language model.

One approach is the "Personalized Transformer for Explainable Recommendation" (PETER), proposed by L. Li et al. (2021). Leveraging the strong

natural language processing capabilities of transformers, they employed this architecture to create personalized explanations. PETER was designed to generate both explanations and recommendation ratings, based on the user ID, item ID, and optional features that serve as model inputs.

The challenge involved assigning meaningful representations to item IDs and user IDs, as traditional transformers treat these IDs similarly to words. To tackle this, they modified the transformer architecture to allow user and item IDs to capture feature information during training. With enough training data, they were able to produce explanations that surpassed state-of-the-art models based on LSTM and GRU technologies (L. Li et al., 2021).

P5 is another approach that relies on the transformer architecture. It was trained on user and item data; therefore, P5 could also handle simple ID's as input. This architecture was able to outperform previously discussed approaches like PETER, Attr2Seq, and NRT in terms of text quality. The model can be considered a general-purpose recommendation model, as it was able to follow a variety of recommender-domain-related instructions, in contrast to models like Attr2Seq and NRT (Geng et al., 2023).

Pre-Trained Transformer-Based Generation

L. Li et al. (2023) proposed a method that relies on a fully trained transformer-based language model, which is an even more generalized approach. Language models are trained on vast amounts of text data, as discussed in Section 2.1.4, enabling them to generate high-quality natural language sentences. Leveraging this capability can enhance the explanation-generation process. They introduced a method called PEPLER-D (Personalized Prompt Learning for Explainable Recommendation, based on discrete prompt learning). The core idea is the construction of a discrete prompt, where user and item IDs are entirely replaced by user and item features (L. Li et al., 2023). The item and user features are available as textual descriptions, which allows language models to understand them semantically. Unlike the previous method, PETER, a pre-trained model, cannot understand IDs without additional information or fine-tuning. This approach has the downside of not being able to encode all information during the training process. The main advantage is that no training data in terms of reviews, users, or items is required. Therefore, the system seems especially suitable for cold-start scenarios (L. Li et al., 2023).

The implementation proposed uses the hugging-face implementation of GPT-2 ¹² and it outperformed previous benchmarks such as text quality metrics and various explainability metrics. Evaluation metrics will be described in more detail in the following section. Figure 2.4 illustrates the basic

¹²<https://huggingface.co/openai-community/gpt2>

structure of the explanation generation process, where user and item IDs are replaced with the actual user and item features already available in the system.

Many similar approaches using pre-trained transformer models have been proposed in recent years. The work of Itallo Silva et al. (2023) can be seen as a proof of concept for recommendations and explanations in the movie domain, where the model is not provided with item feature information. The model solely relies on the general world knowledge captured in the training steps.

Zhou and Joachims (2023) provided a similar approach that used a fully trained GPT model. The survey compared the quality of explanations between human-written explanations and generated ones in the movie domain. Participants did not significantly differentiate between model-generated and human-written texts in terms of ranking movies they hadn't seen before. On the other hand, they rated model-generated texts as better than human-written ones for movies they had seen before. This has led to the conclusion that pre-trained transformer models might be a good source for post-hoc text-based explanations.

Another recent approach by J. Liu et al. (2023) that aims to use GPT as a general purpose recommender for many recommendation tasks found that the P5 model performs better according to text quality metrics, as it was trained based on the explanation ground truth. On the other hand, ChatGPT focuses more on language interaction and diversity, simulating human conversation and considering factors such as context, emotion, and logic. The different design goals and training settings of P5 and ChatGPT contribute to the variation in the generated text (J. Liu et al., 2023). Based on human evaluation, GPT clearly outperformed the P5 model. Figure 2.10 shows an example explanation.

2.3.3. Evaluation of Text-Based Explanations

Evaluating the generated explanations is a fundamental challenge in the field of explainable recommendations. This evaluation judges the model's output, helps improve it, and assesses its performance compared to other models. Over time, various evaluation methods have emerged, some based on natural language quality evaluation and others specifically designed for recommendation explanation.

One intuitive way to evaluate explanation outputs is through case studies, which involve a detailed manual examination of some outputs in specific cases. The main advantage of this method is its intuitiveness, making it simple for readers to grasp what an explanation might look like. Additionally, it is a relatively inexpensive evaluation method, as it does not require

large datasets or a broad and diverse user range for assessment. However, a downside is that it is not feasible to present all computed samples, especially if the training or test set is large. This method makes it difficult to compare results against other models (X. Chen et al., 2022).

For comparison against other models, quantitative measures are highly efficient. Usually, this type of evaluation relies on parametric functions that evaluate the output according to strict rules. The rules could include counting target words, such as features, in the output. Due to its algorithmic nature, this evaluation method can be performed on a large set of data. It usually results in some form of numerical score, which makes it easy to compare against other models (X. Chen et al., 2022).

Popular examples of these metrics are the BLEU and ROUGE scores, which were originally designed for natural language processing (see Section 2.1.7), but also explainability-related metrics like the feature matching ratio, feature coverage ratio, and feature diversity are used (L. Li et al., 2020).

Feature Matching Ratio (FMR). Aims to evaluate the generated explanation on the feature level. It checks if a given feature is included in the generated explanation.

$$FMR = \frac{1}{N} \sum_{u,i} \delta(f_{u,i} \in \hat{S}_{u,i})$$

where $\hat{S}_{u,i}$ is the generated sentence for a user-item pair, $f_{u,i}$ is the given feature, and $\delta(x) = 1$ if x is true and $\delta(x) = 0$ otherwise (L. Li et al., 2020).

Feature Coverage Ratio (FCR). It is used to measure the features at the corpus level, i.e., how many different features are shown in the produced explanations. The ratio gives indications about the number of features the model is actually using to generate explanations.

$$FCR = \frac{N_g}{|F|}$$

where F is the collection of all features in the dataset, and N_g is the number of distinct features shown in the generated explanations (L. Li et al., 2020).

Feature Diversity (DIV). It is reasonable that the explanations for different user-item pairs do not always discuss the same feature, so we are motivated to measure feature diversity. Let $\hat{F}_{u,i}$ and $\hat{F}_{u',i'}$ respectively represent two sets of features contained in two generated explanations, and we can compute

their intersection. For each pair of feature sets corresponding to two explanations in the testing set, the averaged size of the intersection is calculated as:

$$\text{DIV} = \frac{2}{N \times (N - 1)} \sum_{u, u', i, i'} |\hat{F}_{u,i} \cap \hat{F}_{u',i'}|$$

A lower DIV indicates a smaller overlap between feature sets and, thus, a higher diversity. For FMR and FCR, the higher the scores are, the better the performance is (L. Li et al., 2020).

Reviews	Results
Absolutely great product. I bought this for my fourteen year old niece for Christmas and of course I had to try it out, then I tried another one, and another one and another one. So much fun! I even contemplated keeping a few for myself!	Ground Truth: "Absolutely Great Product" P5's output: great colors and great price for the price GPT's output: Love this nail art set - perfect colors and variety
Love the colors. Didn't get any doubles. 1 bottle was not fully closed and the bottle chipped on the neck of the bottle. But being where the break was I just closed it and it is still usable. I wouldn't recommend this for painting your full nail (It is for art), but I would for stamping and nail art. Small brushes great for that. Not all work for stamping though, like the metallic ones.	Ground Truth: "I wouldn't recommend this for painting your full nail (It is for art)" P5's output: "great price and great quality and great price" GPT's output: "SHANY's Nail Art Set is a must-have for creative nails."
Wow, this is the best deal I've seen on nail polish in a long time. You get so many vibrant beautiful colors to choose from. These are nail art brushes for fine detail. I love that you can get a whole kit for this price!	Ground Truth: "this is the best deal I've seen on nail polish in a long time" P5's output: "great price and great quality and great price" GPT's output: "SHANY's Nail Art Set is a must-have for stunning manicures."

Figure 2.10.: Explanation Generation Results for ChatGPT vs. P5 (J. Liu et al., 2023)

Algorithmic evaluation methods do not offer much flexibility, and they usually do not understand the semantics of the text. Figure 2.10 shows an example detailing this issue. The explanation generation model P5 yields a really good performance in terms of text quality based on evaluation methods like BLEU and ROUGE, evaluated on the Beauty¹³ dataset. However, having a closer look at the output shows that, in some cases, the output rarely makes any sense and would not be helpful in the real world. Human evaluation can fix this problem, especially with case studies, crowd-sourcing, or online experiments (J. Liu et al., 2023).

¹³<https://www.kaggle.com/datasets/skillsmuggler/amazon-ratings>

2.3.4. Effects of Explaining Recommendation Items

As discussed in Section 2.3.1, in theory, the goals are clearly defined and impact most of the properties a well-performing recommender system should fulfill. However, it is unclear if users actually appreciate explanations of recommendations. It is important to look at the data and study results to see if the explanations really impact the user's behavior and if the user appreciates the explanations. Even though most research focuses on explanation generation and quality evaluations, some studies have examined their effects on users.

Hoeve et al. (2017) conducted a study focusing on the impact of explanations for news article recommendations. In this study, most news article consumers would like to receive explanations, but there was no clear outcome on how they would like to see them. Also, adding recommendation explanations did not significantly change the open rate for news articles. In fact, many people did not read the explanation.

Tran et al. (2021) conducted research to assess whether users value explanations for recommendations in the movie domain, typically considered low-involvement. Their study primarily focused on determining the circumstances under which users prefer to receive explanations and the explanation types preferred by the users.

The study's design allowed 206 participants to select whether they wanted to see the explanation for a given recommendation item. Resulting in 51% wanting to see the actual explanation. They further found out that the participants are more likely to see the explanation if their satisfaction with the recommended item is lower.

According to the study, participants tend to favor knowledge-based explanations. Because this type of explanation focuses on the user's preferences, it is intuitive to the user and can, therefore, increase their satisfaction.

As movies and news are typically considered low-involvement domains Tran et al. (2023) conducted a study with a focus on users preferences in a variety of domains, containing one low-involvement (restaurant), one medium-involvement (tourism), and one high-involvement (accommodation) domain. Tran et al. (2023) found that users in a high-involvement domain tend to look at the explanation for disliked recommendations to understand the reason for the recommendation. This is quite similar to the findings for the low-involvement domain in the work of Tran et al. (2021). However, the study found that users in low-involvement domains prefer to review the recommendation explanation to determine whether they agree with it.

Further, they found that there is no statistically significant correlation between users with varying characteristics (gender, nationality) and their need for explanations. Depending on the domain, users appreciate different explanation types. For the restaurant domain, users preferred item-based explanations because they got an idea about overall satisfaction. Meanwhile, in the tourism domain, users prefer feature-based explanations. For accommodations, no significant result was found (Tran et al., 2023).

Further research was conducted by Lubos et al. (2024), using a pre-trained LM to generate various explanation types and measure the performance of each generated explanation type compared to well-established baselines. They found that the participants clearly preferred the explanations generated by the LM. Similarly, Silva et al. (2024) found that users were more satisfied with GPT's explanations than other popular sources.

2.3.5. Consequence-Based Explanations

A special form of text-based model-agnostic explanations is the concept of consequence-based explanations for recommendation items proposed in the work of Lubos et al. (2023).

The primary aim of this new concept is to highlight the effects of consuming recommended items, particularly in areas where users lack knowledge about the potential outcomes of these recommendations. These explanations help users to clearly see how their choices affect their circumstances (Lubos et al., 2023). This explanation type is new in the recommender system domain, but it has proven useful in other domains, like reinforcement learning and meteorology.

The American Meteorological Society conducted a study to examine the effects of consequence-based tornado warnings on participants. They concluded that in some instances, these warnings led participants to take more drastic protective measures than they would have without such detailed information (Ripberger et al., 2015).

Further, they were used in the reinforcement learning domain to explain what would happen if an agent behaved in a certain way. This method breaks down complicated model actions into consequences that are easy to understand. It also builds trust and understanding among users by showing why some choices are better than others based on the expected outcomes (van der Waa et al., 2018).

An example from the work of Lubos et al. (2023) is shown to get an idea about the concept of consequence-based explanations.

The number of carbs, sugar, and protein in the cooked meal will give you enough energy for your activity level, and the number of calories and fat in the dish will support you in losing weight.

Furthermore, the explanation should address both the positive and negative effects of a decision, emphasizing the importance of focusing on relevant consequences to avoid overwhelming the user with excessive information. They conducted an online user study to evaluate the effects of consequence-based explanations. The consequences were derived from user preferences and item features; therefore, it is considered a knowledge-based explanation. Further, they focused on the apartment and recipe domains to identify different user behaviors at different levels of involvement, where the apartment domain is considered a high-involvement domain, and the recipe domain is considered low-involvement.

They found that consequence-based explanations positively impact most of the typical explanation goals (Lubos et al., 2023). Explanations based on consequences were more effective and required less time for users to make decisions. Not in the apartment domain, but there were notable differences in the recipe domain. The analysis suggests that highlighting consequences helps users decide faster. It has to be considered that the consequence-based explanations were longer, with around 338 – 749 characters, in comparison to the content-based baseline.

They also found that consequence-based explanations were effective in both low- and high-involvement domains, although the differences in user ratings compared to the content-based explanation baseline were minimal. Overall, user satisfaction was higher in the high-involvement domain than in the low-involvement domain, but they generally positively affect user satisfaction. In terms of transparency, the consequence-based explanations achieved a lower rating compared to the baseline, as they lack insights into how the system generated the suggestion.

2.4. Summary

This chapter has examined key theoretical concepts in the field of recommender systems, addressing common challenges linked to complex recommendation algorithms. The "Black Box" problem is one of these issues, and various technologies, including large language models, are attempting to resolve it by explaining recommendations in a way that is understandable to the user. No single approach solves all the issues and checks all the requirements, resulting in various methods. Explaining recommendations in a text-based way usually involves generating natural language, which is a challenging task in itself. Recent advancements in natural language generation, relying on the transformer architecture, allow for efficient natural language generation.

These advancements are being repurposed in the recommender system domain to find efficient ways to generate explanations. Much research has been conducted to design systems that efficiently generate explanations based on features, demographics, and user data. Consequence-based explanations are a relatively new concept in the recommender system domain, but initial studies in the recommender system domain and other fields have proven their effectiveness. This explanation style emphasizes the impact a recommended item has on the user. Evaluations in the recommender domain have found this explanation style very promising, particularly in effectiveness, efficiency, and user satisfaction. Consequently, finding methods to generate this type of explanation efficiently and with high quality is crucial.

3. Methodology

As there are no established approaches to generate consequence-based explanations efficiently and reliably, this work will use transformer-based language models paired with common prompt engineering techniques to create this kind of explanation. This work's main focus will be exploring how common prompt engineering techniques can be adapted to generate efficient and high-quality explanations. Intending to determine which method is most appropriate and yields the most effective consequence-based explanations.

Research Question: "(1) How can the large language model GPT be leveraged to generate consequence-based explanations for recommendation items?

(2) Does the use of common prompt engineering techniques have an impact on the quality of the explanations?"

Based on the theoretical concepts, descriptions, and related work discussed in the previous chapter, the process of generating consequence-based explanations will be outlined. We begin by defining the relevant domains and introducing a simple yet effective data structure necessary for the generation process. The next step involves selecting appropriate data, relying on the basic concepts of recommender systems. This data will serve as the foundation for the explanation generation using a GPT LM, paired with prompt engineering techniques to improve the explanation quality.

Furthermore, evaluation metrics will be defined, based on the concepts discussed in Section 2.3.3, to assess the quality and performance of the generated explanations. This evaluation will allow for comparing different prompting methods regarding explanation quality, performance, and cost. The explanations should focus primarily on the consequences derived from the user profile and the item properties. The explanations should address both positive and negative impacts, tailor their formulation to the user, maintain consistency, avoid hallucinations, and adhere to the length restrictions for each domain.

3.1. Domains and Data

The generation of explanations is highly dependent on the chosen domains and the data structure. For the scope of this work, we perform the generation for one high-involvement and one low-involvement domain. Like the work of Lubos et al. (2023), we stick to the apartment- and the recipe-domain.

As discussed in Section 2.3.5, the consequences are based on item features. Therefore, we must choose a suitable, common, and realistic data structure representing them to generate the explanations. The base format of the data should be as realistic as possible so that it could potentially mirror a real-world application dataset. Additionally, we create detailed user profiles with fields similar to the item feature fields. Typically, recommender systems play a key role in creating these profiles. However, for this work, we assume the availability of detailed user profiles since the focus is on post-hoc explanation generation.

We assume that the data is available in comma-separated value (CSV) format. We generate 50 user profiles and 50 items per domain for the sake of variety in the dataset. Below, the actual fields for each domain are defined.

The data for both domains was generated synthetically using the GPT-3.5 language model (Ouyang et al., 2022), and was manually reviewed to ensure consistency and plausibility.

3.1.1. Data Information: Apartment Domain

In contrast to the work of Lubos et al. (2023), we decided to create a more detailed dataset for the apartment domain based on the assumption that more feature information will result in a broader range of consequence descriptions. Table 3.1 shows the fields of the apartment data.

Apartment-Item Features	User-Preferences
itemid	userid
rent_price	rent_price
distance_to_center	distance_to_center
size	size
rooms	rooms
bathrooms	bathrooms
furnished	furnished
kitchen_appliances	kitchen_appliances
private_parking	private_parking
outdoor_space	outdoor_space
leisure_facilities	leisure_facilities

Table 3.1.: Apartment Item fields and User Preference fields

3.1.2. Data Information: Recipe Domain

For the dataset in the recipe domain, we chose similar fields to the work of Lubos et al. (2023), with some minor modifications, for simplicity and structuring reasons.

Recipe Feature	User Feature
itemid	userid
title	-
ingredients	ingredients
cuisine_type	cuisine_type
difficulty	difficulty
diet	diet
cooking_time	cooking_time
calories_per_100g	calories_per_100g
fat_per_100g	fat_per_100g
carbs_per_100g	carbs_per_100g
protein_per_100g	protein_per_100g
allergenic	allergenic

Table 3.2.: Matching Recipe Features with User Features

The features in this domain are commonly found in most real-world applications. Therefore, it is reasonable to generate explanations based on this data.

The item fields are nearly identical to the user preference fields. This has the benefit of simple feature matching when creating the user-item pairs.

3.2. Evaluation Metrics

Possible ways of evaluating natural language and recommendation explanations were discussed in Sections 2.3.3, 2.1.7. Using quantitative metrics will allow us to compare prompt engineering techniques based on a broad range of user-item input pairs. As previously seen, it can be important to evaluate natural language using some human evaluation-based metrics too. Therefore, we also selected an evaluation method based on a case study, primarily focusing on specific examples, to gain deeper insights into the structure of the generated explanations.

In this work, we aim to introduce a distinct approach to generating explanations that diverge significantly from existing methods in the field. Consequently, drawing direct comparisons between our approach and others,

such as Attr2Seq, NRT, or PEPLER-D, may not be relevant. We anticipate that our method will yield results that are not directly comparable to those obtained by traditional techniques.

3.2.1. Quantitative Metrics

As shown in Section 2.3.3, various metrics can be used to assess the text quality and explanation quality of explanations. For the scope of this work, it is not necessary to evaluate the text quality based on metrics like the BLEU or ROUGE score. Due to the use of GPT models, we assume that the text is of high quality and close to the results from evaluations that were already performed in the work of OpenAI (2023) and Brown et al. (2020). The more important metrics for this work are feature- and explainability-based measures listed below:

- Average Explanation Length
- User Matching Ratio
- Feature Coverage Ratio
- Feature Diversity

These metrics will be evaluated based on the results of all the mentioned prompting strategies. Due to the ambiguities and synonyms inherent in natural language, the metrics (see Section 2.3.3) require a slight adaptation for our applications. The specific algorithm adaptations are discussed below, but they will offer similar information about explanations as the original implementation.

User Matching Ratio The feature matching ratio is designed to work for algorithms that generate explanations based on a single input feature. Therefore, we have to slightly adapt the feature matching ratio to fit the needs of our explanation generation. Since we are generating the explanation based on multiple features, we want to evaluate how well it is tailored to the user and its properties.

Text embeddings are used to compute the user matching ratio. The user profile and the explanation are encoded into an embedding vector. Then, the vectors are compared using cosine similarity, giving us a value between 0 and 1. If the cosine similarity is close to 0, the explanation generally does not match the user profile. A value close to 1, on the other hand, indicates that the explanation talks about most of the user features. This ratio will provide insight into how effectively the explanations align with user preferences.

Feature Coverage (adapted) The original implementation of this metric displays the ratio of features that the model uses to generate the explanation. Due to the high performance of GPT, in our case, the ratio would be close to 100% since the model mentions all of the features across the entire explanation text corpus. Therefore, it is more interesting to see if the model prioritizes some features or neglects others. For this evaluation, we count the features' occurrences across all explanations. As a result, we get an occurrence metric based on the feature level, which allows us to see how the model prioritizes the features.

To count features, we create an embedding for the text of each feature and compare it with the embedding of the explanation text. A resulting high cosine similarity indicates that a feature is mentioned in the explanation. Text embeddings make this comparison much more flexible and, therefore, more robust to semantics. For instance, if the feature is "outdoor space: garden" and the model refers to the outdoor space as "back yard", a simple text-matching approach would not be able to capture this. In contrast, the embedding approach will successfully identify the match.

Feature Diversity As earlier stated, we already have a method to determine if the explanation mentions a feature. Computing the set of features for each explanation gives us an indication of the number of different features that are mentioned per feature. Further, we can compute the DIV (see Section 2.3.3), which provides information about the overlap in the feature sets, indicating the variety of features discussed on the explanation level.

3.2.2. Case-Study

We perform the case study on specific output prompts to compare the explanations generated by the model and observe the impact of the prompt engineering steps on individual explanation examples. To evaluate some of the results manually, we need a definition of quality criteria derived from the information and findings of related work described in previous chapters.

- **Consequence-Based Sentence Style.** Is the explanation focusing on consequences?
- **User-Targeted Explanation.** Is the formulation of the explanation user-targeted, reasonable, and suitable for the task and the domain?
- **Hallucination-Free.** Is the reasoning behind the consequences correct?

In the case study, we will look at an average explanation example for each domain. Further, we examine an example where the user offers no flexibility (e.g., `recipe.allergenic = Gluten AND user.allergenic = Gluten`). We expect

the model to identify the issue and, ideally, provide a solution. Additionally, we will examine a sample that produced particularly brief explanations, as it is likely that these explanations convey very few consequences.

3.3. Implementation Details

This section defines the environment for creating explanations. Initially, a language model and the corresponding configuration parameters must be defined. Moreover, we will briefly discuss the recommendation algorithm that produces the user-item pairs.

3.3.1. Use of GPT

To generate the explanations, we rely on a GPT model. As discussed in the Section 2.1.7, they are among the SOTA language models for reasoning and instruction following, and therefore, they will be a decent fit for the experiments. There are various implementations of GPT models that are either ready to use on a local machine or accessible via the Rest API. Our experiments' setup ensures low API costs and eliminates the need for complex model adaptations. Consequently, ready-to-use models over API calls work best for the experiments.

OpenAI offers an API that provides access to various models, each with distinct strengths and weaknesses. These include capabilities in multimodality, text embeddings, and a diverse range of customization and parameterization options. On the official website of OpenAI ¹, all the current versions of GPT models are listed and briefly described. The most effective one for our experiments has its strength in reasoning and instruction following. The context-window size does not matter, considering that the text input and the generated output are rather short.

The requirements leave us with two models: gpt-4-1106-preview (GPT-4) and gpt-3.5-turbo-1106 (GPT-3.5). As detailed in Section 2.1.7, GPT-4 models yield better results, especially on reasoning datasets like "Hella Swag", but the generation cost is 20x as high as with GPT-3.5. Additionally, the GPT-3.5 model is the fastest one; consequently, it would be most suitable to generate explanations in real-world applications. Therefore, we chose GPT-3.5 to run the generation.

¹<https://platform.openai.com/docs/models/>

GPT API (Python) and Configuration

OpenAI² provides an API ³ that offers a wide range of language model capabilities, including access to various GPT models and word-embedding functionality.

For the explanation generation, we need one single endpoint of the GPT API.

```
completion = CLIENT.chat.completions.create(  
    model="gpt-3.5-turbo",  
    max_tokens=150,  
    temperature=0,  
    messages=messages  
)
```

The code block demonstrates an example use of the GPT API with the Python library provided by OpenAI.

- **model:** This parameter selects the specific GPT model. In our case, it will be `gpt-3.5-turbo`.
- **max_tokens:** This parameter sets a hard limit on the generation length of the model. By instructing the model to generate a specific number of tokens, there is a risk that the model might exceed this amount. Additionally, the model does not “see” this configuration, so we must instruct it to produce a concise response and use `max_tokens` as a safeguard.
- **temperature:** A value between 0 and 2 determines the level of “randomness” in the model’s token selection process. A low value, in theory, makes the output more predictable and deterministic. Unfortunately, full determinism is impossible with the current system design (see Section 2.1.8).
- **messages:** This parameter provides all the inputs, instructions, and chat history based on which the model generates its output.

The roles indicate to the model which text was generated by GPT, which was system-generated (e.g., persona prompt), and which inputs are user-generated. Here is a short example of the `messages` object in the GPT API, demonstrating the use of the `system`, `user`, and `assistant` roles:

²<https://openai.com/>

³<https://platform.openai.com/docs/api-reference/introduction>

```
messages = [
    {
        "role": "system",
        "content": "You are a helpful assistant."
    },
    {
        "role": "user",
        "content": "Can you explain how recommender systems work?"
    },
    {
        "role": "assistant",
        "content": "Sure! Recommender systems ... "
    }
]
```

In this example:

- The system role sets the behavior and context of the assistant.
- The user role provides the input or question from the user.
- The assistant role is the generated response from GPT.

3.3.2. Recommender System and User-Item Pair Generation

The focus goes towards the generation of explanations, so there is no need for a complex and sophisticated recommendation algorithm to create user-item pairs. We perform the mapping using a content-based matching approach, computing a match score for each item feature and corresponding user preference. The greater the number of item features that align with user preferences, the higher the match score will be.

In contrast to the work of Lubos et al. (2023), there are no item features that strictly eliminate a potential user-item pair.

In real-world scenarios, it would make sense to eliminate the pairs where the user does not have any flexibility (e.g., User Diet = Vegan and Recipe \neq Vegan). However, given our interest in the explanations, observing how GPT-3.5 addresses these types of scenarios will be interesting. Appendix A shows some examples of user-item pairs.

For the quantitative analysis, we have generated 50 user-item pairs for each domain based on a high match score. Consequently, we will have 50 distinct explanations per domain, which is the basis for the quantitative evaluation.

3.3.3. Data Preprocessing

We must preprocess the data and establish a predefined input structure. The user and the item will be represented using a JSON ⁴ structure. This format is commonly used in real-world applications nowadays and has much less overhead than similar structured formats like XML. Therefore, it will be a suitable fit for our user and item representation.

The ID fields are unnecessary for generating the explanation, as they provide no useful information to the pre-trained model we use, similar to L. Li et al. (2023) (PEPLER-D). The GPT model would treat them as mere words without semantic meaning. In the Prompt 3.1, a complete example of an item-json and a user-json can be seen.

3.4. Consequence-based Explanation: Baseline

Establishing a baseline is a critical step in the model development and evaluation process, particularly when exploring innovative concepts like consequence-based explanations. Usually, baselines serve as a reference point to track the performance and improvements of changing systems. As consequence-based explanations are a relatively new concept, there are no baselines available to track the performance of the explanation generation. This section provides details and information about the baseline, which will serve as the “lower bound” for the evaluation.

3.4.1. Baseline generation

The baseline is a naive approach to generate explanations without proper domain knowledge or advanced, prompt engineering steps. It just relies on the already available data in the form of generated user-item pairs. The generation is based on one simple instructive prompt that includes the user preferences and the item features, along with a short instruction that details the output we expect from the model. In addition, we have to add a length restriction to the prompt so that the model is aware of the restricted length. Prompt 3.1 shows the prompt used to generate the baseline output for the user-item pair.

⁴ECMA-404: JSON Standard

Baseline Prompt

User: Provide a detailed explanation of the consequences for the user when selecting this apartment item. Ensure to address how the choice of this item impacts the user and detail any potential outcomes. Please do not exceed 100 words.

User Preferences:

```
{
  "rent_price": "850",
  "distance_to_center": "0.7",
  "size": "40",
  "rooms": "1",
  "bathrooms": "1",
  "furnished": "fully",
  "kitchen_appliances": "electric stove",
  "private_parking": "false",
  "outdoor_space": "none",
  "leisure_facilities": "bars, cafes"
}
```

Apartment Item:

```
{
  "rent_price": "850",
  "distance_to_center": "0.7",
  "size": "45",
  "rooms": "1",
  "bathrooms": "1",
  "furnished": "fully",
  "kitchen_appliances": "electric stove, microwave",
  "private_parking": "false",
  "outdoor_space": "none",
  "leisure_facilities": "bars - 0.1km,
                        art galleries - 0.4km"
}
```

Prompt 3.1.: Baseline Prompt, that shows the full example of a user-json and an item-json

3.5. Prompt Engineering Techniques

Besides the model configuration, prompt engineering is the only point where we can have an impact on the model's output performance. Consequently, this section focuses on applying advanced and well-tested prompt engineering methods (see Section 2.1.6), modifying the baseline input prompt so that the output improves and aligns with the properties that our consequence-based explanations should fulfill. The prompt engineering steps will be applied to the same user-item pair as used in the baseline (see Table A.1).

3.5.1. Generate a Persona

Defining a "persona" can be beneficial for the performance of the language model as it contains a general description of consequence-based explanations so that the model is aware of the concepts, the environment, and the expected behavior of the model when answering user input prompts.

First, we must create a short but pregnant description of consequence-based explanations. As discussed in Section 2.3.5, consequence-based explanations should highlight the positive and negative impacts of consuming the recommended item. Based on the description, we create a prompt that provides the model with all the necessary information about consequence-based explanations.

Persona Definition

System:

Assume the persona of a [domain_placeholder] domain expert. Your task is to explain to a user why a recommended item is suitable for him. Your explanation should not focus just on matching features. It should focus on the positive and negative impacts on the user if they choose the item. Ensure that the explanation is detailed and tailored to the user's needs.

Note that it is important to keep the persona definition short and pregnant. It should contain the necessary information but no complex tasks.

Over the Rest API, this persona definition can be passed to the GPT model with the role of "System" which tells the model that this is a precondition that holds for the following request.

For the final request, we reuse the instruction from the baseline prompt.

3.5.2. Chain-of-Thought Approach

As detailed in Section 2.1.6, breaking down a long and complex prompt into multiple smaller prompts can increase the model's performance, especially for reasoning tasks. Each step generates knowledge that the model can reuse to ultimately construct a solution for the overall problem.

In the context of explanation generation, we split the task into two sub-tasks. Initially, the model should evaluate the consequences of consuming an item with specified properties. In the next step, the model should formulate an explanation based on the consequences that were previously generated.

Due to the task's simplicity, we will proceed with the generation of consequences without defining a persona.

In the second step, we will provide the complete chat history in which the model created the consequences and add an instruction prompt. The instruction prompt tells the model to provide a consequence-based explanation based on the consequences discussed earlier in the chat, tailored to a specific user. Note that the complete "chat history" is submitted with each request to keep up the "chain-of-thought". The example below shows a shortened example running this technique with inputs and outputs.

Chain-Of-Thought Conversation

User:

Describe the consequences and impacts on a person who lives in an apartment with the following properties. Do not exceed 70 words.

User:

Property: rent_price: 850

Assistant:

Living in an apartment with rent price of 850 may lead to financial strain and reduced disposable income for the resident. This could limit their ability to save, invest, or enjoy leisure activities, impacting their overall quality of life. Limited financial resources may also cause stress and anxiety, affecting mental well-being. Additionally, it could restrict their housing options or force them to compromise on other essential expenses.

Prompt 3.2.: Chain-of-Thought Prompt 1/3

Chain-Of-Thought Conversation

User:

Property: distance_to_center: 0.7

Assistant:

Living in an apartment with a distance to the city center of 0.7 may result in convenience and accessibility to amenities, workplaces, and entertainment options. The resident could benefit from saving time and money on commuting, enjoying a vibrant urban lifestyle, and having easy access to public transportation. This proximity may enhance their social life, work-life balance, and overall well-being, offering a convenient and desirable living situation.

...

User:

Property: leisure_facilities: bars - 0.1km, art galleries - 0.4km

Assistant:

Living in an apartment with leisure facilities such as bars and art galleries located at close distances (0.1km and 0.4km respectively) may offer convenient access to entertainment and cultural experiences for the resident. They can easily explore nightlife, socialize, and engage in artistic activities within walking distance. This proximity may enhance their leisure options, social life, and overall lifestyle by providing diverse opportunities for recreation and enrichment.

User:

Assume the persona of an apartment domain expert. Your task is to explain to a user why a recommended item is suitable for him. Your explanation should not focus just on matching features, your explanation should focus on the positive and negative impacts on the user if they choose the item. Ensure that the explanation is detailed and tailored to the user's needs. Create a consequence-based explanation that emphasizes what impacts the user can expect from selecting this apartment item. Consider the previously discussed consequences to generate the explanation that describes the impacts. Do not exceed 100 words.

Prompt 3.3.: Chain-of-Thought Prompt 2/3

Chain-Of-Thought Conversation

User Preferences:

```
{
  "rent_price": "850",
  ...
  "leisure_facilities": "bars, cafes"
}
```

Apartment Item:

```
{
  "rent_price": "850",
  ...
  "leisure_facilities": "bars - 0.1km,
                        art galleries - 0.4km"
}
```

Prompt 3.4.: Chain-of-Thought Prompt 3/3, shortened for better readability

The actual implementation of this method is done by simply creating a message-object and adding an instructive prompt:

```
Describe the consequences and impacts on a person
who lives in an apartment with the following
properties. Do not exceed 70 words.
```

A feature description will be appended to the message object (e.g., "Property: rent price: 1300"), and the message object will be submitted to the GPT API. Then, GPT generates the consequences for the previously submitted property. This is done for each feature so that our message object finally contains all the properties and the corresponding consequence responses. The final prompt is then added to the message object, instructing the model to reuse the created consequences to create a full consequence-based explanation.

3.5.3. External Knowledge-Base Approach

As briefly discussed in 2.1.6, all of our approaches can be seen as "external knowledge base prompting". Similar to the approach of (L. Li et al., 2023), we have to provide textual user and item information for each request. Technically, this information is sourced from an external knowledge base. But this section takes the external knowledge-base approach one step further. To apply the external knowledge-base (external-kb) approach to our specific use case, we leverage the general world knowledge that GPT has already captured. GPT already has all the information needed to generate

explanations based on consequences. Therefore, we can reuse GPT as our "external knowledge-base". Similar to the "chain-of-thought" approach, we use GPT to generate consequences for the item properties. But this time, we collect all of them, and later on, when generating an explanation, we provide all the important consequences directly in the prompt.

The core concept here is to "interrupt" the chain-of-thought and provide the model with the required information in a single request. This reduces the complexity and length of the request while still including similar content. Further, this method is scalable because it significantly reduces the number of requests, which is particularly advantageous when generating explanations for a big audience. Prompt 3.6 includes similar information as Prompt 3.4 but in the chain-of-thought approach, $|\text{item-features}| + 2$ requests are made, while the external-kb approach only needs a single request.

External Knowledge-Base

System:

Assume the persona of an apartment domain expert. Your task is to explain to a user why a recommended item is suitable for him. Your explanation should not focus just on matching features, your explanation should focus on the positive and negative impacts on the user if they choose the item. Ensure that the explanation is detailed and tailored to the user's needs.

User:

User Preferences:

```
{
  "rent_price": "850",
  ...
  "leisure_facilities": "bars, cafes"
}
```

Apartment Item:

```
{
  "rent_price": "850",
  ...
  "leisure_facilities": "bars - 0.1km, art galleries - 0.4km"
}
```

Prompt 3.5.: External Knowledge-Base 1/2, shortened for better readability

External Knowledge-Base

Consequences:

Living in this apartment will offer financial stability since the rent matches your budget. Its proximity to the city center provides convenience, though it may come with noise and congestion. The larger size offers more comfort, but could mean higher utilities and maintenance. The fully furnished space saves time, though limits personalization. The nearby bars enhance social options but could be noisy, while art galleries add cultural value and relaxation opportunities.

Create a consequence-based explanation that emphasizes what impacts the user can expect from selecting this apartment item. Consider the provided consequences to generate the explanation that describes the consequences. Do not exceed 100 words.

Prompt 3.6.: External Knowledge-Base 2/2, shortened for better readability

3.5.4. Few-Shot Approach

Few-shot prompting is one of the most common and well-known prompting methods. It is also one of the most promising ones.

For both domains, we create example input-output combinations that, later on, will be prepended to the actual input baseline prompt, including the persona definition. The input-output samples are created manually. Three samples are randomly chosen from the collection of sample input-output combinations. Prompt 3.8 shows an example prompt.

Few-Shot Prompt

System:

Assume the persona of an apartment domain expert. Your task is to explain to a user why a recommended item is suitable for him. Your explanation should not focus just on matching features. Your explanation should focus on the positive and negative impacts on the user if they choose the item. Ensure that the explanation is detailed and tailored to the user's needs.

Prompt 3.7.: Few-Shot Prompt 1/3, first "shot"

Few-Shot Prompt

User:

User Preferences:

```
{  
  "userid": "user1",  
  ....  
  "leisure_facilities": "gym, park"  
}
```

Apartment Item:

```
{  
  "itemid": "item2",  
  ...  
  "leisure_facilities": "gym - 0.5km, park - 0.2km"  
}
```

Create a detailed explanation of the consequences of selecting this apartment item for the user. Please do not exceed 100 words.

Assistant:

Choosing this apartment will offer a lower rent price, allowing you to save money. However, its proximity to the city center means you might experience noise during the day and evening. The studio meets your requirements in terms of size, rooms, and bathroom. It is partially furnished, so you'll need to purchase some additional furniture. The apartment also features a balcony, providing a space to enjoy the sun and spend your free time outside. Leisure facilities are conveniently nearby, with the gym just 0.5 kilometers away and the park only 0.2 kilometers away. Overall, you may need to compromise on the level of furnishing for the convenience of being closer to the city center, having access to leisure facilities, and saving money.

[Few-shot example 2]

[Few-shot example 3]

Prompt 3.8.: Few-Shot Prompt 2/3, first "shot"

Few-Shot Prompt

User:

User Preferences:

```
{  
  "rent_price": "850",  
  ...  
  "leisure_facilities": "bars, cafes"  
}
```

Apartment Item:

```
{  
  "rent_price": "850",  
  ...  
  "leisure_facilities": "bars - 0.1km, art galleries - 0.4km"  
}
```

Provide a detailed explanation of the consequences for the user when selecting this apartment item. Ensure to address how the choice of this item impacts the user and detail any potential outcomes. Please do not exceed 100 words.

Prompt 3.9.: Few-Shot Prompt 3/3, shortened for better readability

4. Results and Discussion

This chapter provides a detailed look at the evaluation results of the generated explanations. It contains quantitative evaluation results and a case study. The discussion details how the results can be interpreted and what conclusions we draw from the explanation-generation results.

4.1. Results

4.1.1. Quantitative Analysis

Prompt Size

First, we have to look at a metric to compare the cost of each method. The cost of a prompt is usually determined by the token size/character amount of each input prompt. The following chart shows the average length in characters for each prompting strategy:

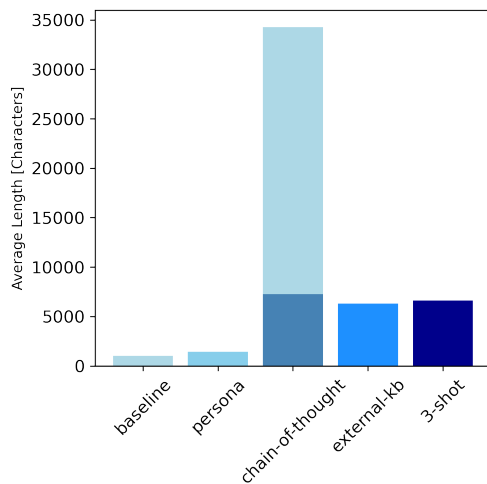


Figure 4.1.: Apartment Domain: Prompt length per strategy

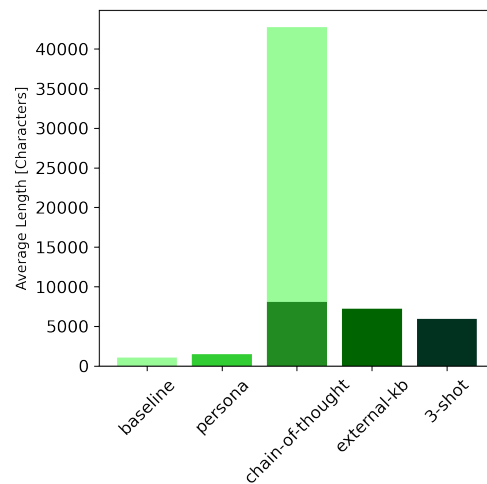


Figure 4.2.: Recipe Domain: Prompt length per strategy

The chain-of-thought approach is built upon multiple requests, where the dark-colored part of the bar corresponds to the final request to generate the explanation, and the light-colored part represents the character inputs until

the final prompt can be created. The figures show that the chain-of-thought approach is the most expensive for both domains. In contrast, the external-kb and few-shot approaches are significantly shorter. The persona and baseline approaches are the most cost-effective, with the persona approach being slightly larger due to the length of the persona description.

Average Explanation Length

Figure 4.3 shows the average length of the explanations grouped by the prompting strategy. The maximum length for the apartment domain is 100 words, and the maximum length for the recipe domain is 60 words. In the few-shot domain, all examples used as "shots" adhere to the length restrictions.

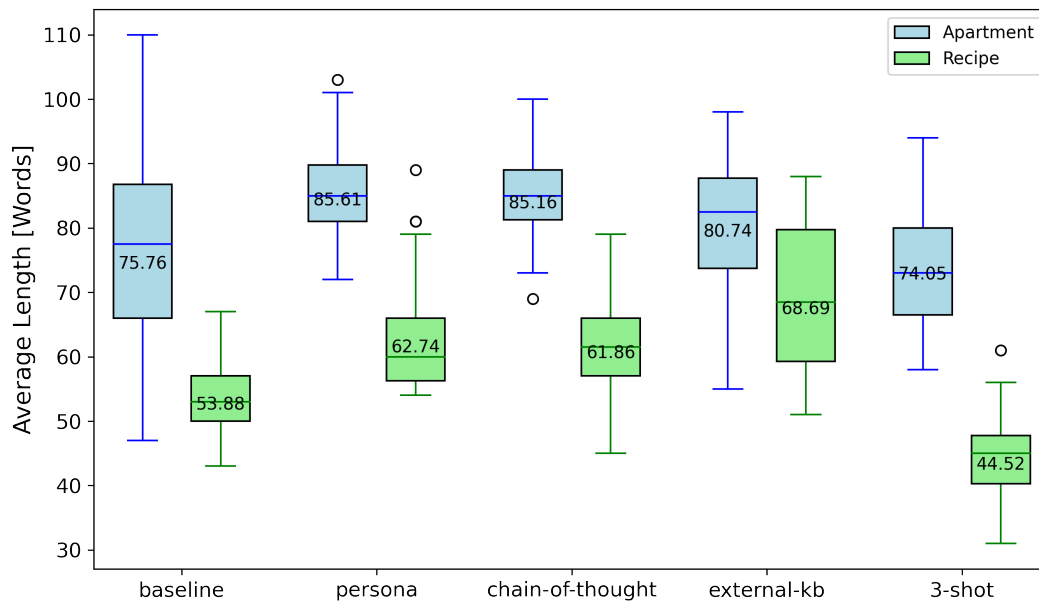


Figure 4.3.: Explanation length grouped by prompting strategies

In the recipe domain, the persona, chain-of-thought, and external-kb approaches exceed the maximum length. In contrast, only a few outliers in the baseline and persona strategies surpass the defined limit in the apartment domain.

User Matching Ratio

By comparing the explanation to the user's properties, the user matching ratio reveals how well the explanation fits the user. Computing the average across all user-item pairs provides a more detailed view of the different prompting strategies compared to the baseline.

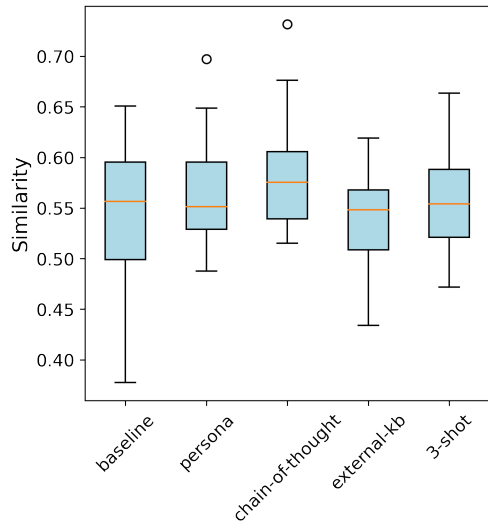


Figure 4.4.: Apartment Domain: User Matching Ratio

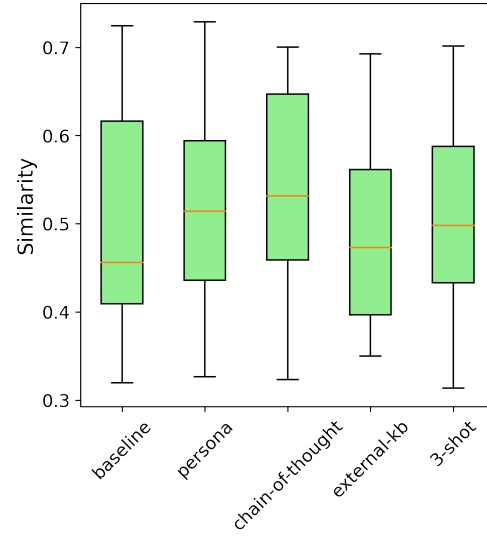


Figure 4.5.: Recipe Domain: User Matching Ratio

As shown in Figure 4.4, the matching ratios are quite similar, except for the chain-of-thought approach, which results in a significantly higher similarity. This suggests a more personalized explanation that better aligns with the user's preferences.

The recipe domain (see Figure 4.5) shows a similar result, as the chain-of-thought approach is the best-performing one, followed by the persona approach.

Feature Coverage

This metric provides insight into the features discussed in the explanations, indicating which ones the model considers important. Further, it shows the differences between the prompting strategies and the feature distribution. The full explanation dataset consists of 50 explanations per domain.

Below, the statistics for the apartment domain (blue) and the recipe domain (green) are shown:

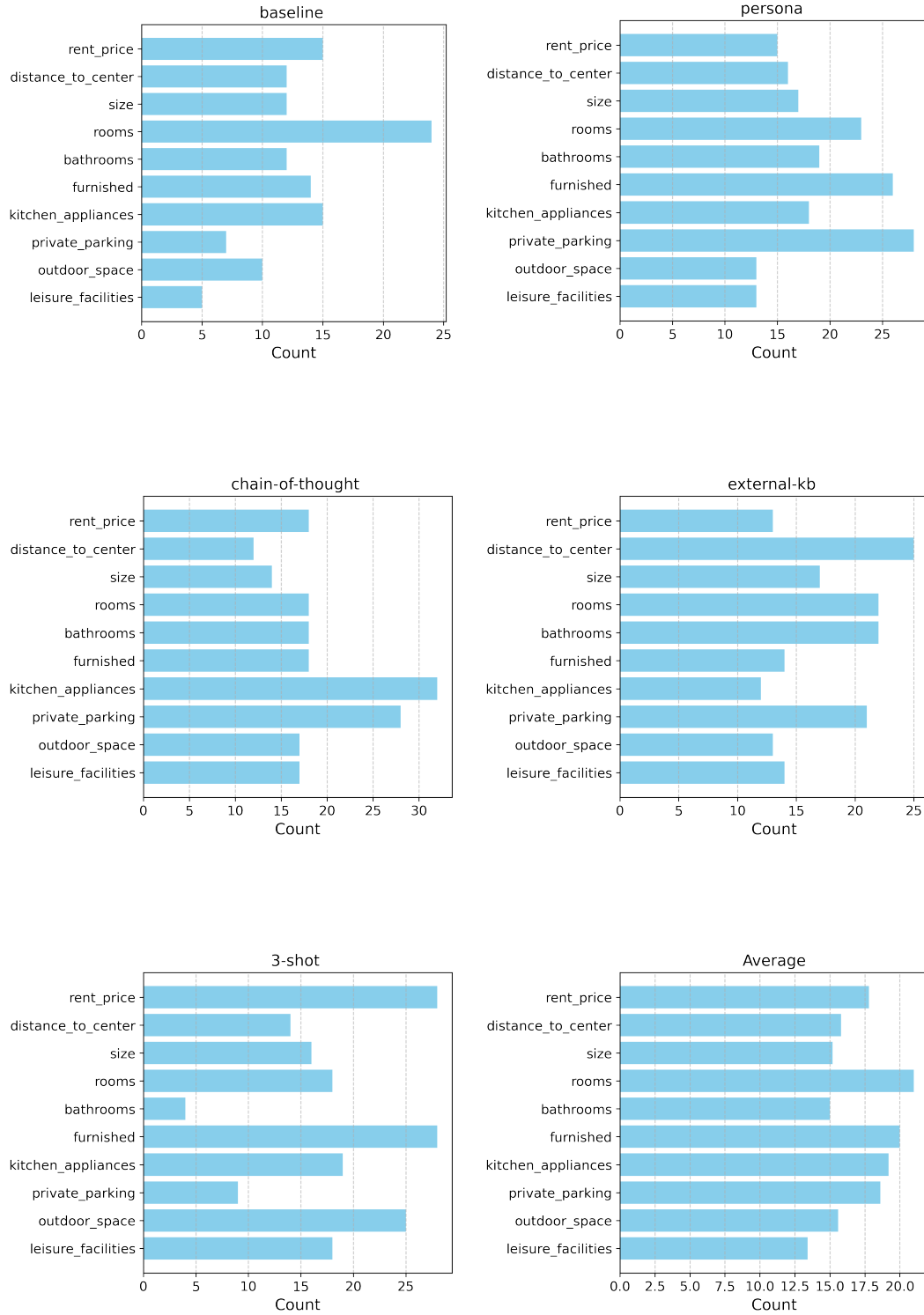


Figure 4.6.: Feature coverage across consequences for the apartment domain, illustrating the number of features discussed in the explanations and highlighting the features considered important by the model.

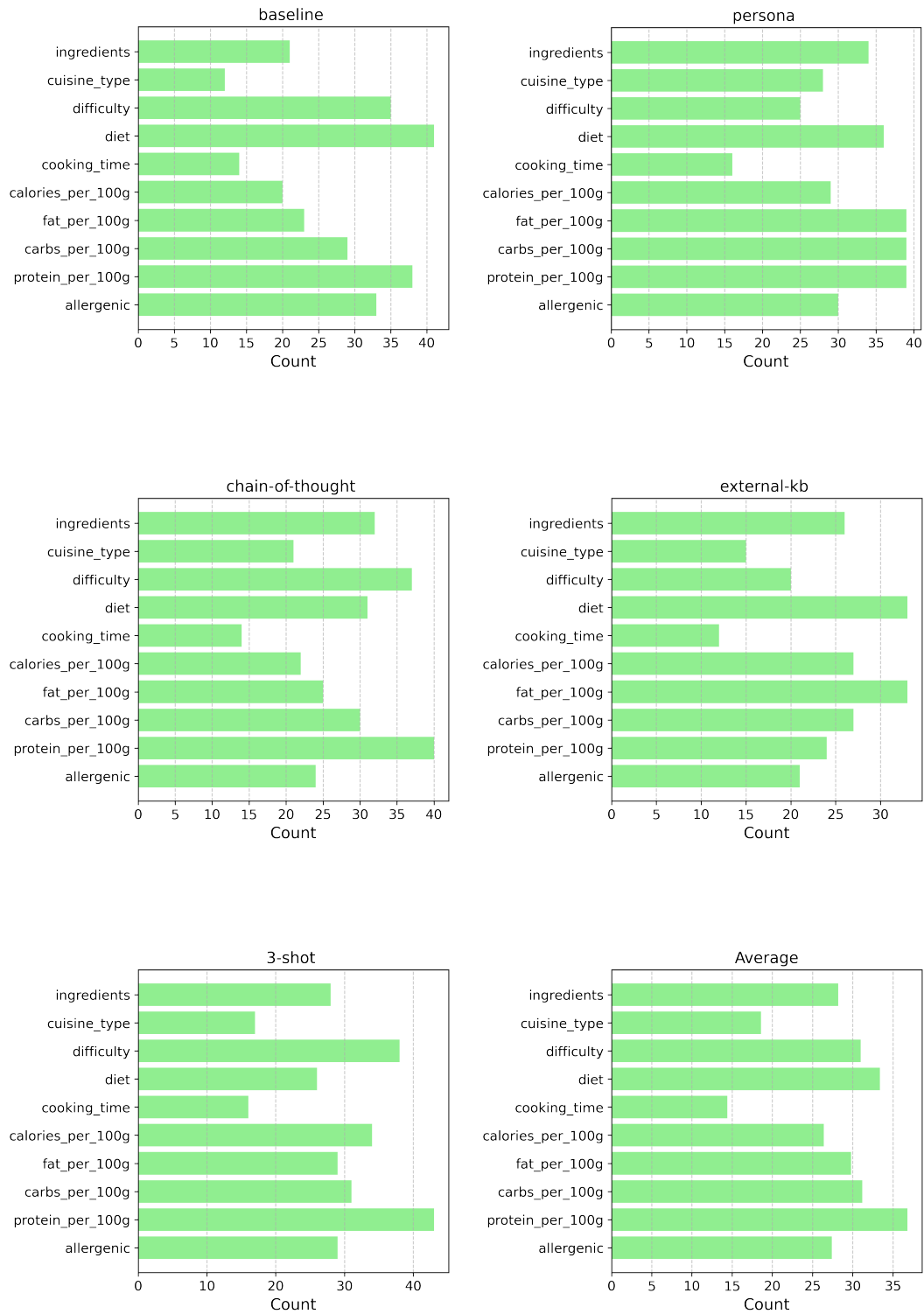


Figure 4.7.: Feature coverage across consequences for the apartment domain, illustrating the number of features discussed in the explanations and highlighting the features considered important by the model.

The persona, chain-of-thought, and external-kb feature coverage in the apartment domain (Figure 4.6) exhibits a relatively even distribution of features, with only a few outliers. The 3-shot and baseline results are relatively skewed. The features that, on average, were most mentioned are "rent_price", "rooms", "furnished", "kitchen_appliances", "private_parking". Features that are considered less important are "leisure_facilities", "bathrooms", and "size".

The results are different for the recipe domain (Figure 4.7). All the distributions exhibit more significant skewness compared to the apartment domain. The persona approach shows the most uniform distribution in the recipe domain. Important features are "protein_per_100g", "diet", "difficulty" and "carbs_per_100g". Features considered less important are "cooking_time" and "cuisine_type".

Feature Diversity

The feature diversity allows us to monitor the versatility of feature selection across all explanations. This can be an indication of high-quality explanations. A low value indicates high diversity, and a high value indicates low diversity.

	Apartment	Recipe
Baseline	1.2105	2.3363
Persona	2.5561	3.0568
Chain-of-Thought	2.7240	2.3895
External-KB	2.1507	1.7725
Few-Shot	2.5405	2.6636

Table 4.1.: Diversity score grouped by explanation technique and domain with the lowest diversity score (bold) indicating the highest feature diversity

Table 4.1 shows, that the baseline and the external-kb approach are best performing in terms of feature variety.

The above-computed data also allows us to compute the average number of features mentioned per explanation. This can give further indications about the quality of the explanations.

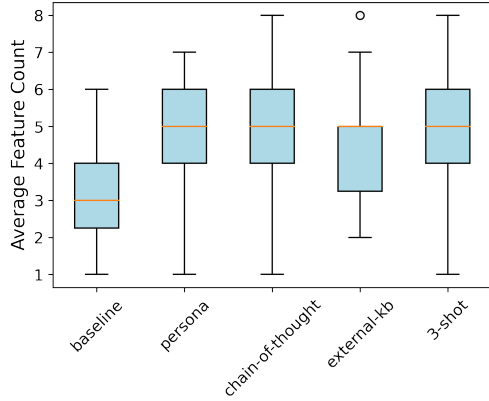


Figure 4.8.: Apartment Domain: Average Feature Count

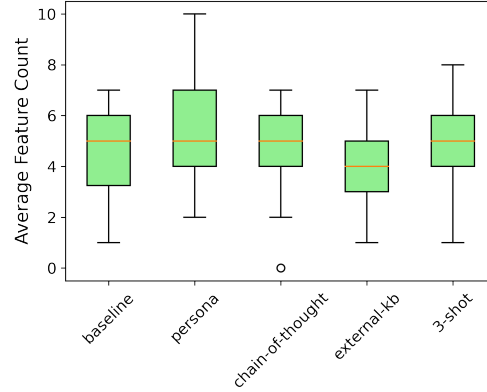


Figure 4.9.: Recipe Domain: Average Feature Count

Figure 4.8 shows that the average number of features in the apartment domain is around five for all prompting strategies, with the baseline explanations mentioning two fewer features on average.

In the recipe domain (Figure 4.9), the external-kb approach mentions an average of four features, trailing behind all other strategies, which mention five features.

4.1.2. Case Study - Qualitative Analysis

Having a close look at the explanations and evaluating them based on the metrics defined in Section 3.2.2 will give us further information about the quality of the explanations for each strategy. We will primarily look at the consequence-focus of the explanation, evaluate if the formulation is suitable for an explanation scenario, and check if the reasoning behind the explanations and consequences is correct.

In Appendix A.1, all the user-item pairs used for this evaluation are listed. The blue underlined text indicates the consequences.

Apartment Domain: Average Example

The following summarizes the analysis of the generated explanations for an average example in the apartment domain, shown in Table 4.2.

- **Consequence-Based Sentence Style:**

- baseline: focuses on the features rather than consequences, mostly describing why the item is a suitable fit for the user based on the features.
- persona: includes some consequences derived from features but is also quite focused on the features.

Prompt	Output
Baseline	Selecting this apartment item may result in the user having a slightly larger living space with an additional kitchen appliance, but they may also miss out on having a microwave. <u>The proximity to leisure facilities may be more favorable, with bars and art galleries within walking distance.</u> However, the user will not have access to private parking or outdoor space. Overall, the user <u>may enjoy a more convenient lifestyle in terms of leisure activities, but may need to consider parking and outdoor space options.</u>
Persona	Choosing this recommended apartment will offer you a slightly larger living space (45 sqm) with improved kitchen appliances including a microwave, <u>which can enhance your cooking experience.</u> The close proximity to bars (0.1km) and art galleries (0.4km) aligns with your preference for leisure facilities. However, the <u>absence of private parking might pose a challenge if you own a vehicle.</u> Overall, <u>this option caters to your main priorities within your budget, offering convenience and entertainment options nearby .</u>
Chain of Thought	Based on your preferences, the recommended apartment aligns closely with your needs. While slightly larger than desired, the fully furnished unit <u>offers convenience.</u> The proximity to bars and art galleries <u>provides easy access to entertainment options.</u> However, non-existent outdoor space may limit relaxation opportunities. Electric kitchen appliances <u>could impact your cooking preferences ,</u> and <u>parking challenges might arise.</u> Overall, this apartment <u>offers convenience and entertainment but may pose limitations in outdoor activities and cooking choices.</u>
External KB	Choosing this apartment means you'll enjoy a slightly larger space for comfortable living . The location close to the city center offers <u>easy access to amenities but may come with noise and higher costs.</u> The additional microwave provides <u>cooking convenience,</u> though you'll still be limited without a stove. <u>Nearby bars can tempt excess drinking, affecting your well-being.</u> The art galleries <u>can inspire creativity and cultural appreciation.</u> Overall, this choice <u>offers convenience and cultural opportunities, but watch out for potential noise disturbances and lifestyle impacts .</u>
Few Shot	Selecting this apartment will provide the additional benefit of having a microwave in the kitchen. The apartment also offers more space than initially desired, <u>which could potentially result in slightly higher utility costs.</u> Additionally, the proximity to bars and art galleries may <u>lead to increased noise levels in the area, impacting the overall tranquility of the living environment.</u>

Table 4.2.: Apartment Domain: Example explanations for user-item pair (Table A.2). The text highlighted in blue indicates the consequences.

-
- chain-of-thought: increased focus towards consequences, but the consequences are rather vague.
 - external-kb: is deriving a lot of consequences from the features, focusing mostly on consequences, and the consequences are more detailed than with the chain-of-thought strategy
 - few-shot: while this example includes just three consequences, they are quite detailed.
- **User-Targeted Explanatory Formulation:**
 - baseline: fails to directly target the user as a potential reader.
 - Others: the other approaches correctly target the explanation to the user. The formulation of each is suitable for an explanation scenario.
 - **Hallucination-Free:**
 - external-kb: is affected by a hallucination, as the apartment item has a stove, and the explanation was stating "... though you'll still be limited without a stove."
 - Others: No other hallucinations were found.

The baseline approach provided a reasonable explanation, though it is primarily content-based and lacks detailed consequences. Similarly, the persona approach yielded comparable results but was able to address the user directly.

Overall, the external-kb approach was the most detailed regarding consequences, with the chain-of-thought method also offering comprehensive information about consequences. Conversely, the promising few-shot approach produced a disappointing result because of the low number of consequences mentioned.

Recipe Domain: Average Example

Prompt	Output
Baseline	Choosing the Chicken Caesar Salad may result in excessive calorie and fat intake due to the high fat content (25g per 100g) when compared to the user’s moderate preference. Additionally, the presence of allergens such as gluten, dairy, and eggs may pose risks for those with allergies or intolerances. It’s essential to consider these factors before selecting this recipe item.
Persona	Choosing the Chicken Caesar Salad may not align perfectly with your initial ingredient preferences but offers a tasty alternative. While it provides a good source of protein, the high fat content and allergens present (gluten, dairy, egg) may not suit everyone. It’s relatively quick and easy to prepare, making it a convenient option with a moderate calorie count.
Chain of Thought	Choosing the Chicken Caesar Salad aligns with your non-vegetarian diet preference and easy cooking style. However, be cautious due to its allergenic ingredients (gluten, dairy, egg) which may trigger adverse reactions if you have allergies. The salad offers moderate calories but high fat content, so be mindful of portion sizes for a balanced diet and consider any dietary restrictions or health concerns before consuming.
External KB	Choosing the Chicken Caesar Salad may have negative impacts on your health. This dish is high in unhealthy fats, cholesterol, and allergens like gluten and dairy, which can lead to weight gain, heart disease, and digestive issues. Opting for a recipe with healthier fats from salmon and asparagus would better support your overall health and wellness.
Few Shot	Selecting the Chicken Caesar Salad would not fully align with your preferences for non-vegetarian, high protein, and high-fat dishes like salmon and asparagus. This dish contains allergens like gluten, dairy, and eggs, which may not suit your needs or dietary restrictions.

Table 4.3.: Recipe Domain: Example explanations for user-item pair (Table A.3). The text highlighted in blue indicates the consequences.

The following summarizes the analysis of the generated explanations for an average example in the recipe domain, shown in Table 4.3.

- **Consequence-Based Sentence Style:**
 - baseline: mostly focusing on the features rather than consequences
 - persona: mostly focusing on features

-
- chain-of-thought: increased focus towards consequences, with more detailed consequences in comparison to persona and baseline
 - external-kb: similarly focused on consequences as chain-of-thought
 - few-shot: includes one vague consequence, focused on a content-based style
 - **User-Targeted Explanatory Formulation:**
 - baseline: fails to directly target the user as a potential reader.
 - Others: the other approaches correctly target the explanation to the user. The formulation of each is suitable for an explanation scenario.
 - **Hallucination-Free:**
 - All: No hallucinations found.

The results are similar to those in the apartment domain, with the external-kb and chain-of-thought approaches focusing mainly on consequences. However, most of the consequences of the external-kb approach are negative.

Apartment Domain: Short Average Explanation Length

This example shows the user-item pair with the shortest average explanation length, with an average of 68 words. The baseline is the overall shortest explanation in this example, with 53 words.

Prompt	Output
Baseline	Selecting this apartment item may result in the user having a slightly longer distance to the center (0.7km instead of 0.5km) but gaining an additional kitchen appliance (microwave) and closer access to leisure facilities like bars and a grocery store. The user may benefit from <u>added convenience and amenities, potentially enhancing their overall living experience</u> .
Persona	Choosing the recommended apartment will offer you a slightly larger living space (50 sqm) closer to the city center (0.5km). The addition of a microwave to the kitchen appliances <u>will add convenience</u> . Though lacking private parking, the proximity to a grocery store (0.1km) <u>will ease your shopping</u> . The presence of nearby bars (0.3km) aligns with your leisure preferences. Overall, this choice <u>enhances your living experience with added convenience and proximity</u> , making it a suitable option for your needs.
Chain of Thought	The recommended apartment balances your rent price requirement while offering additional space compared to your preferred size. The convenience of fully furnished living <u>may enhance comfort</u> but slightly exceeds your desired distance from the center. Although lacking outdoor space, it is equipped with leisure facilities nearby. The electric stove and microwave <u>provide efficient cooking options</u> , and close proximity to a grocery store <u>complements your lifestyle</u> . However, the <u>absence of private parking might pose occasional challenges</u> . Overall, <u>this choice aligns well with your preferences, offering a blend of comfort, convenience, and accessibility</u> .
External KB	Choosing this apartment <u>will offer you a convenient lifestyle with easy access to grocery stores for fresh produce and daily essentials</u> . However, it's important to note that the close proximity to bars <u>may lead to noise disturbances and temptations for excess drinking, impacting your sleep quality and overall well-being</u> . Additionally, the limited outdoor space and lack of private parking <u>may cause inconveniences and potential security risks</u> . Be prepared for potential noise disturbances and consider how important outdoor space and parking access are to you.
Few Shot	Selecting this apartment will offer the same rent price within your budget and slightly larger space. The closer distance to the city center <u>allows for easier access to your preferred leisure facilities</u> , with bars just 0.3 km away and a grocery store nearby. The addition of a microwave in the kitchen provides extra convenience for meal preparation. However, the lack of outdoor space <u>may limit your options for relaxation</u> .

Table 4.4.: Apartment Domain: Example explanations for user-item pair (Table A.4)
. This pair generates shorter than average explanations. The text highlighted in blue indicates the consequences.

The following summarizes the analysis of the generated explanations for an example with shorter than average explanations in the apartment domain, shown in Table 4.1.2.

- **Consequence-Based Sentence Style:**
 - baseline: lacks specific consequences; it primarily focuses on explaining features.
 - persona: minor indications about consequences, focus goes towards features.
 - chain-of-thought: more consequence-based, but the consequences are not that detailed
 - external-kb: mostly focused on a detailed explanation of some consequences
 - few-shot: the low amount of consequences and consequences are not detailed
- **User-Targeted Explanatory Formulation:**
 - baseline: fails to directly target the user as a potential reader.
 - Others: the other approaches correctly target the explanation to the user. The formulation of each is suitable for an explanation scenario.
- **Hallucination-Free:**
 - All: No hallucinations found.

As anticipated, the results are similar to the average example, with the notable difference that fewer consequences are mentioned. The external-kb approach performed the best, followed by the chain-of-thought approach.

Recipe Domain: Bad User-Item Match

For this user-item pair, we expect the model to notify the user, as the user should not consume this item, because of the allergenic. (recipe.allergenic = user.allergenic = Gluten).

The following is a summary of the analysis of the generated explanations for an example with a significant mismatch between user profile and item features in the recipe domain, shown in Table 4.5.

- baseline: identified the issue, but was not able to express it correctly and user-targeted.
- persona: correctly identified the problem and recommended a solution.
- chain-of-thought: correctly identified and notified user.
- external-kb: no identification of the problem.
- few-shot: correctly identified the problem.

The persona, chain-of-thought, and few-shot examples were closest to the expected behavior, with the persona approach being able to identify the problem and suggest a solution.

The explanations have similar issues and advantages to those already discussed in the average example 4.3 where the external-kb approach and the chain-of-thought approach are closest to the requirements of consequence-based explanations.

Prompt	Output
Baseline	Selecting the Pulled Pork Sandwich recipe item may have consequences for the user, as it is high in calories, fat, carbs, and protein. The sandwich is also allergenic to gluten, which could affect individuals with gluten sensitivities. <u>Users may need to consider portion sizes and balance their meal with lighter options to maintain a healthy diet.</u>
Persona	Choosing the Pulled Pork Sandwich aligns with your preferences for American cuisine and non-vegetarian diet. However, be mindful that each 100g serving contains high calories, fat, carbs, and protein. <u>If consumed in moderation, it can be a delicious treat, but overindulgence may impact weight and cholesterol levels.</u> Given your gluten allergy, opt for gluten-free buns.
Chain of Thought	Considering your preferences for a non-vegetarian, American dish with medium difficulty, the Pulled Pork Sandwich fits your bill. However, the high calorie, fat, carb content <u>might impact your diet negatively.</u> With a high protein content, it can assist muscle growth . Being gluten allergenic, <u>it may lead to adverse effects if you have gluten sensitivities.</u> Moderation is crucial for a balanced approach.
External KB	Opting for the Pulled Pork Sandwich <u>could lead to health risks since it contains high levels of unhealthy fats, sugars, and sodium.</u> Consistent consumption <u>may result in weight gain, high cholesterol, increased heart disease risk, and reduced energy levels.</u> Choosing this dish frequently <u>could lead to chronic health issues and a diminished quality of life.</u> Be mindful of your health choices.
Few Shot	Selecting the Pulled Pork Sandwich aligns perfectly with your non-vegetarian, high protein, high-fat diet. However, the high carb content may not meet your preferences. Additionally, the presence of buns <u>might be an issue due to your gluten allergy .</u>

Table 4.5.: Recipe Domain: Example explanations for user-item pair (Table A.5). The text highlighted in blue indicates the consequences.

4.2. Discussion

This work aimed to identify effective methods to generate consequence-based explanations relying on large language models. The research addressed the following question: "(1) How can the large language model GPT be leveraged to generate consequence-based explanations for recommendation items?

(2) Does the use of common prompt engineering techniques have an impact on the quality of the explanations?"

The quantitative analysis suggests that the GPT model can identify features and formulate explanations based on the features, as found in L. Li et al. (2023). Even the baseline prompting method was capable of doing so. The chain-of-thought approach had minor advantages in some of the metrics (user matching ratio and feature coverage), indicating that it is beneficial for the model to break down the explanation generation into smaller parts, similar to the findings from Wei et al. (2023).

Interestingly, the external-kb approach yielded quite different results from the chain-of-thought approach, even though they rely on similar input data. Only the input structuring is different, which could indicate that the "thought chain" positively impacts the models' performance.

The few-shot and persona approaches yielded explanations similar in performance to the baseline across most metrics. This is surprising given that the few-shot approach has proven its outstanding capabilities in other domains OpenAI (2023) (see Figure 2.5).

This outcome appears to shift in the qualitative evaluation, as the baseline cannot focus on consequences, and the explanation was not appropriately directed to the user. Upon examining the persona approach, it became clear that the user-targeting issue had been resolved, suggesting that the baseline lacked some contextual information. Nevertheless, the persona approach did not really focus on the consequences either.

In contrast, the chain-of-thought approach focused mostly on consequences rather than just pointing out the features. Similarly, the external-kb prompting generates explanations mostly based on very detailed consequences. This could be the reason for the rather low user matching ratio, as it mainly discusses consequences rather than the underlying features that are the main factor for a high user matching ratio. These methods effectively focused on the broader impacts of an item rather than merely listing features. This further supports the idea that breaking the explanation process into smaller tasks — like in the chain-of-thought approach — enables the model to infer consequences better.

The few-shot approach yielded poor qualitative results, likely due to shorter explanations and limited examples ("shots"). Additionally, the model may have focused too much on input features presented in the examples, resulting in a skewed feature distribution.

Given the resource-intensive nature of prompting, developing an external knowledge base that organizes potential consequences in a structured format would be more efficient, as the current chain-of-thought approach is not practical for large-scale applications. Persona prompting, on the other hand, offers a cost-effective alternative. While it may not fully emphasize consequences, it could be viable when the explanation doesn't require a strong consequence-focus. The baseline model, as implemented in this study, is not suitable for real-world applications since it lacks essential domain-specific knowledge and broader contextual understanding.

(1) Overall, the explainability metrics indicate that GPT can efficiently generate explanations with sufficient context, especially provided in the form of a persona. However, there are limitations when focusing specifically on consequence-based explanations. The model is not really capable of focusing on consequences just by instructing it to do so. These limitations could be addressed by further refining the prompts.

(2) Prompt engineering techniques significantly influence the quality of consequence-based explanations. Simple instructive prompts often lack the necessary context and information, leading to suboptimal, non-consequence-focused results. The model performs better with more detailed information and context, and its performance improves further when it has access to consequence-specific data, as in the chain-of-thought and external-kb approaches.

5. Conclusion and Future Work

The work has shown common prompt engineering techniques to generate consequence-based explanations for recommendation items. The resulting explanations were evaluated based on various metrics to identify differences in terms of explanation quality and performance. The GPT model identified features in which the reader could be interested and generated highly personalized explanations. While all the prompting strategies could generate personalized explanations, only two could meet the requirement to create consequence-based explanations. When relying on prompt engineering techniques such as Chain-of-Thought or External Knowledge-Base prompting, the results have shown the GPT models' capabilities to correctly identify consequences based on the input and generate correct and concise explanations out of them. Since the Chain-of-Thought approach is not practical for large-scale applications requiring real-time explanation generation, the External Knowledge-Base approach, which also focuses on consequences, is more suitable for such applications and yields similar results.

Future Work

The evaluations performed in this work gave some general insights into how useful and suitable the GPT model is for generating consequence-based explanations. Still, a user study would provide valuable insights into user preferences and the effectiveness of different explanations in future research.

Further, the quantitative metrics do not provide information about the explanations' consequence-focus. In the case of the user matching ratio, the strategy of the External Knowledge-Base performed below average due to its focus on highly detailed consequences, which the evaluation metrics could not capture. A metric measuring the consequence-focus could solve this problem in future research. For example, a "transition word count" metric could be valuable, as consequences are frequently signaled by terms such as "therefore," "moreover," and similar words.

Another area that requires attention is the optimal length for consequence-based explanations. These explanations tend to be longer than classical knowledge-based explanations due to the detailed description of consequences. Therefore, future research should focus on determining the ideal length that balances comprehensiveness and readability without creating an overload of information that causes users to disengage.

Alternative designs and implementations can also be evaluated, especially for the few-shot and chain-of-thought-promoting strategies. The chain-of-thought approach, as defined in this study, is not suited for real-time, large-scale platforms. Further, in this study, the few-shot approach produced poor explanation results compared to other fields, where it was among the best-performing approaches. Future research could experiment with a range of "shots" and more versatile and detailed explanation examples. These adaptations could improve the performance of the few-shot approach.

Addressing these limitations and exploring the proposed future research directions can enhance the understanding and effectiveness of consequence-based explanations in various applications.

Bibliography

- Balog, K., & Radlinski, F. (2020). Measuring recommendation explanation quality: The conflicting goals of explanations. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 329–338. <https://doi.org/10.1145/3397271.3401032> (cit. on p. 30).
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Nee-lakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020). Language models are few-shot learners. (Cit. on pp. 11, 15–17, 48).
- B.Thorat, P., Goudar, R., & Barve, S. (2015a). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110, 31–36. <https://doi.org/10.5120/19308-0760> (cit. on pp. 24, 25).
- B.Thorat, P., Goudar, R., & Barve, S. (2015b). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110, 31–36. <https://doi.org/10.5120/19308-0760> (cit. on p. 25).
- Chen, C., Zhang, M., Liu, Y., & Ma, S. (2018). Neural attentional rating regression with review-level explanations. *Proceedings of the 2018 World Wide Web Conference*, 1583–1592. <https://doi.org/10.1145/3178876.3186070> (cit. on p. 34).
- Chen, H., Chen, X., Shi, S., & Zhang, Y. (2021). Generate natural language explanations for recommendation. (Cit. on pp. xii, 31, 33, 34, 36).
- Chen, X., Zhang, Y., & Wen, J.-R. (2022). Measuring “why” in recommender systems: A comprehensive survey on the evaluation of explainable recommendation. <https://arxiv.org/abs/2202.06466> (cit. on pp. 30, 39).
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. <https://arxiv.org/abs/1406.1078> (cit. on p. 36).
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). Training verifiers to solve math word problems. (Cit. on p. 13).

-
- de Gemmis, M., Lops, P., Musto, C., Narducci, F., & Semeraro, G. (2015). Semantics-aware content-based recommender systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (pp. 119–159). Springer US. https://doi.org/10.1007/978-1-4899-7637-6_4 (cit. on p. 24).
- Dong, L., Huang, S., Wei, F., Lapata, M., Zhou, M., & Xu, K. (2017, April). Learning to generate product reviews from attributes. In M. Lapata, P. Blunsom, & A. Koller (Eds.), *Proceedings of the 15th conference of the European chapter of the association for computational linguistics: Volume 1, long papers* (pp. 623–632). Association for Computational Linguistics. <https://aclanthology.org/E17-1059> (cit. on p. 36).
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2024). Retrieval-augmented generation for large language models: A survey. (Cit. on p. 14).
- Geng, S., Liu, S., Fu, Z., Ge, Y., & Zhang, Y. (2023). Recommendation as language processing (rlp): A unified pretrain, personalized prompt predict paradigm (p5). <https://arxiv.org/abs/2203.13366> (cit. on p. 37).
- Goodman, B., & Flaxman, S. (2017). European union regulations on algorithmic decision making and a “right to explanation”. *AI Magazine*, 38(3), 50–57. <https://doi.org/10.1609/aimag.v38i3.2741> (cit. on p. 28).
- Gunawardana, A., & Shani, G. (2015). Evaluating recommender systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (pp. 265–308). Springer US. https://doi.org/10.1007/978-1-4899-7637-6_8 (cit. on pp. 26, 27).
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735> (cit. on p. 5).
- Hoeve, M., Heruer, M., Odijk, D., Schuth, A., Spitters, M., & de Rijke, M. (2017). Do news consumers want explanations for personalized news rankings. <https://api.semanticscholar.org/CorpusID:777073> (cit. on p. 41).
- Itallo Silva, Alan Said, Leandro Balby Marinho, & Martijn Willemsen. (2023). Leveraging Large Language Models for recommendation and Explanation. *Joint Workshop on Interfaces and Human Decision Making for Recommender Systems* (cit. on p. 38).
- Jariha, P., & Jain, S. K. (2018a). A state-of-the-art recommender systems: An overview on concepts, methodology and challenges. *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 1769–1774. <https://doi.org/10.1109/ICICCT.2018.8473275> (cit. on p. 23).
- Jariha, P., & Jain, S. K. (2018b). A state-of-the-art recommender systems: An overview on concepts, methodology and challenges. *2018 Second*

-
- International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 1769–1774. <https://doi.org/10.1109/ICICCT.2018.8473275> (cit. on p. 23).
- Khusro, S., Ali, Z., & Ullah, I. (2016). Recommender systems: Issues, challenges, and research opportunities. In K. J. Kim & N. Joukov (Eds.), *Information science and applications (icisa) 2016* (pp. 1179–1189). Springer Singapore. (Cit. on p. 27).
- Konstan, J., & Riedl, J. (2012). Recommender systems: From algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22, 101–123. <https://doi.org/10.1007/s11257-011-9112-x> (cit. on p. 22).
- Li, L., Zhang, Y., & Chen, L. (2020). Generate neural template explanations for recommendation. *CIKM* (cit. on pp. 39, 40).
- Li, L., Zhang, Y., & Chen, L. (2021). Personalized transformer for explainable recommendation. *ACL* (cit. on pp. 36, 37).
- Li, L., Zhang, Y., & Chen, L. (2023). Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems (TOIS)* (cit. on pp. 15, 37, 53, 58, 77).
- Li, P., Wang, Z., Ren, Z., Bing, L., & Lam, W. (2017). Neural rating regression with abstractive tips generation for recommendation. *SIGIR* (cit. on p. 36).
- Liddy E.D. (2001). Natural Language Processing. In *Encyclopedia of Library and Information Science*. 2nd Ed. NY. Marcel Decker, Inc. (Cit. on p. 4).
- Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 74–81. <https://aclanthology.org/W04-1013> (cit. on p. 16).
- Liu, J., Liu, C., Zhou, P., Lv, R., Zhou, K., & Zhang, Y. (2023). Is chatgpt a good recommender? a preliminary study. <https://arxiv.org/abs/2304.10149> (cit. on pp. xii, 16, 38, 40).
- Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. (Cit. on p. 6).
- Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., Wu, Z., Zhu, D., Li, X., Qiang, N., Shen, D., Liu, T., & Ge, B. (2023). Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. (Cit. on p. 19).
- Lubos, S., Tran, T. N. T., Erdeniz, S. P., Mansi, M. E., Felfernig, A., Wundara, M., & Leitner, G. (2023). Concentrating on the impact: Consequence-based explanations in recommender systems. (Cit. on pp. 1, 42, 43, 46, 47, 52).
- Lubos, S., Tran, T. N. T., Felfernig, A., Polat Erdeniz, S., & Le, V.-M. (2024). Llm-generated explanations for recommender systems. *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and*

-
- Personalization*, 276–285. <https://doi.org/10.1145/3631700.3665185> (cit. on pp. 1, 42).
- Musto, C., Gemmis, M. d., Lops, P., Narducci, F., & Semeraro, G. (2022). Semantics and content-based recommendations. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (pp. 251–298). Springer US. https://doi.org/10.1007/978-1-0716-2197-4_7 (cit. on p. 24).
- Naili, M., Chaibi, A. H., & Ben Ghezala, H. H. (2017). Comparative study of word embedding methods in topic segmentation [Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France]. *Procedia Computer Science*, 112, 340–349. <https://doi.org/https://doi.org/10.1016/j.procs.2017.08.009> (cit. on p. 24).
- Nikolakopoulos, A. N., Ning, X., Desrosiers, C., & Karypis, G. (2021). Trust your neighbors: A comprehensive survey of neighborhood-based methods for recommender systems. <https://arxiv.org/abs/2109.04584> (cit. on pp. 24, 25).
- Ochmann, J., Zilker, S., & Laumer, S. (2021). The evaluation of the black box problem for ai-based recommendations: An interview-based study (cit. on p. 28).
- OpenAI. (2023). Gpt-4 technical report. (Cit. on pp. xii, 1, 8, 16–19, 48, 77).
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. (Cit. on pp. xii, 8–11, 19, 46).
- Pal, K., & Patel, B. (2020). Computer linguistics for processing human language for artificial intelligence – hard applications. 10, 152–159 (cit. on p. 3).
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 311–318. <https://doi.org/10.3115/1073083.1073135> (cit. on p. 16).
- Patel, B., Desai, P., & Panchal, U. (2017). Methods of recommender system: A review. 2017 *International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 1–4. <https://doi.org/10.1109/ICIIECS.2017.8275856> (cit. on p. 23).
- Pauls, A., & Klein, D. (2011). Faster and smaller n-gram language models. *Proceedings of the 49th annual meeting of the Association for Computational Linguistics: Human Language Technologies*, 258–267 (cit. on p. 4).
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training (cit. on pp. 1, 5, 7–9).

-
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. Retrieved January 6, 2023, from <https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405ccod6169988371b2755e573cc28650d14dfe> (cit. on p. 11).
- Rahdari, B., Ding, H., Fan, Z., Ma, Y., Chen, Z., Deoras, A., & Kveton, B. (2024). Logic-scaffolding: Personalized aspect-instructed recommendation explanation generation using llms. *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. <https://doi.org/10.1145/3616855.3635689> (cit. on p. 13).
- Ricci, F., Rokach, L., & Shapira, B. (2022). Recommender systems: Techniques, applications, and challenges. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (pp. 1–35). Springer US. https://doi.org/10.1007/978-1-0716-2197-4_1 (cit. on pp. 1, 21, 22, 25–27).
- Ripberger, J. T., Silva, C. L., Jenkins-Smith, H. C., & James, M. (2015). The influence of consequence-based messages on public responses to tornado warnings. *Bulletin of the American Meteorological Society*, 96(4), 577–590. <https://doi.org/10.1175/BAMS-D-13-00213.1> (cit. on p. 42).
- Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1), 59. <https://doi.org/10.1186/s40537-022-00592-5> (cit. on pp. 27, 28).
- Silva, Í., Marinho, L., Said, A., & Willemsen, M. C. (2024). Leveraging chatgpt for automated human-centered explanations in recommender systems. *Proceedings of the 29th International Conference on Intelligent User Interfaces*, 597–608. <https://doi.org/10.1145/3640543.3645171> (cit. on p. 42).
- Song, W., Wang, C., & Ning, K. (2021). Generate personalized explanations for recommendation based on keywords. *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 4, 51–57. <https://doi.org/10.1109/IMCEC51613.2021.9482221> (cit. on p. 32).
- Srivastava, M., Arora, S., & Boneh, D. (2024). Optimistic verifiable training by controlling hardware nondeterminism. <https://arxiv.org/abs/2403.09603> (cit. on p. 20).
- Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. *Proc. Interspeech 2012*, 194–197. <https://doi.org/10.21437/Interspeech.2012-65> (cit. on p. 4).
- Tintarev, N., & Masthoff, J. (2007). A survey of explanations in recommender systems. *2007 IEEE 23rd International Conference on Data Engineering Workshop*, 801–810. <https://doi.org/10.1109/ICDEW.2007.4401070> (cit. on pp. 30, 31).
- Tintarev, N., & Masthoff, J. (2022). Beyond explaining single item recommendations [Publisher Copyright: © Springer Science+Business Media,

-
- LLC, part of Springer Nature 2011, 2015, 2022.]. In F. Ricci, L. Rokach, B. Shapira, & P. Kantor (Eds.), *Recommender systems handbook* (3rd ed., pp. 711–756). Springer. https://doi.org/10.1007/978-1-0716-2197-4_19 (cit. on pp. 1, 29, 30).
- Tran, T. N. T., Felfernig, A., Le, V. M., Chau, T. M. N., & Mai, T. G. (2023). User needs for explanations of recommendations: In-depth analyses of the role of item domain and personal characteristics. *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*, 54–65. <https://doi.org/10.1145/3565472.3592950> (cit. on pp. 30, 41, 42).
- Tran, T. N. T., Le, V. M., Atas, M., Felfernig, A., Stettinger, M., & Popescu, A. (2021). Do users appreciate explanations of recommendations? an analysis in the movie domain. *Proceedings of the 15th ACM Conference on Recommender Systems*, 645–650. <https://doi.org/10.1145/3460231.3478859> (cit. on pp. xiv, 30, 32, 41).
- Turner, R. (2023, April). An introduction to transformers. (Cit. on p. 8).
- van der Waa, J., van Diggelen, J., van den Bosch, K., & Neerincx, M. (2018). Contrastive explanations for reinforcement learning in terms of expected consequences. <https://arxiv.org/abs/1807.08706> (cit. on p. 42).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023a). Attention is all you need. <https://arxiv.org/abs/1706.03762> (cit. on pp. 3, 5, 7, 8).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023b). Attention is all you need. <https://arxiv.org/abs/1706.03762> (cit. on p. 6).
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models. (Cit. on pp. xii, 13, 77).
- White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. (Cit. on p. 12).
- Xu, L., Zhang, J., Li, B., Wang, J., Cai, M., Zhao, W. X., & Wen, J.-R. (2024). Prompting large language models for recommender systems: A comprehensive framework and empirical analysis. (Cit. on p. 12).
- Yang, A., Wang, N., Cai, R., Deng, H., & Wang, H. (2022). Comparative explanations of recommendations. *Proceedings of the ACM Web Conference 2022*. <https://doi.org/10.1145/3485447.3512031> (cit. on p. 33).
- Yao, L., & Guan, Y. (2018). An improved lstm structure for natural language processing. *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, 565–569. <https://doi.org/10.1109/IICSPI.2018.8690387> (cit. on p. 5).

-
- Zhang, Y., & Chen, X. (2020). Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1), 1–101. <https://doi.org/10.1561/15000000066> (cit. on pp. 29, 30).
- Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, 83–92. <https://doi.org/10.1145/2600428.2609579> (cit. on p. 33).
- Zhou, J., & Joachims, T. (2023). Gpt as a baseline for recommendation explanation texts. <https://arxiv.org/abs/2309.08817> (cit. on p. 38).

Appendix

Appendix A.

Appendix 1: User-Item Pair examples

Key	Apartment Feature	User Feature
rent_price	850	850
distance_to_center	0.7	0.7
size	45	40
rooms	1	1
bathrooms	1	1
furnished	fully	fully
kitchen_appliances	electric stove, microwave	electric stove
private_parking	false	false
outdoor_space	none	none
leisure_facilities	bars - 0.1km, art galleries - 0.4km	bars, cafes

Table A.1.: Apartment Domain: Example user-item pair used to demonstrate the prompt engineering steps

Key	Apartment Feature	User Feature
id	item21	user15
rent_price	850	850
distance_to_center	0.7	0.7
size	45	40
rooms	1	1
bathrooms	1	1
furnished	fully	fully
kitchen_appliances	electric stove, microwave	electric stove
private_parking	false	false
outdoor_space	none	none
leisure_facilities	bars - 0.1km, art galleries - 0.4km	bars, cafes

Table A.2.: Apartment Domain: Example user-item pair used for the case study "Apartment Domain: Average Example"

Key	Item Feature	User Feature
id	item42	user10
ingredients	Romaine Lettuce, Croutons, Parmesan Cheese, Caesar Dressing, Chicken	Salmon, Asparagus, Lemon, Olive Oil
cuisine_type	Global	Global
difficulty	Easy	Easy
diet	Non-Vegetarian	Non-Vegetarian
cooking_time	20 min	20 min
calories_per_100g	350	moderate
fat_per_100g	25g	high
carbs_per_100g	10g	low
protein_per_100g	25g	high
allergenic	Gluten, Dairy, Egg	-

Table A.3.: Recipe Domain: Example user-item pair used for the case study "Recipe Domain: Average Example"

Key	Item Feature	User Feature
id	item46	user2
rent_price	1400	1200
distance_to_center	0.8	1.2
size	80	60
rooms	2	2
bathrooms	1	1
furnished	partially	partially
kitchen_appliances	electric stove, large fridge	electric stove
private_parking	true	false
outdoor_space	balcony	balcony
leisure_facilities	bus stop - 0.1km, restaurants - 0.2km	N/A

Table A.4.: Apartment Domain: Example user-item pair that produces shorter than average explanations

Key	Item Feature	User Feature
id	item15	user15
ingredients	Pork Shoulder, BBQ Sauce, Buns, Coleslaw	Pork Shoulder, BBQ Sauce, Buns, Coleslaw
cuisine_type	American	American
difficulty	Medium	Medium
diet	Non-Vegetarian	Non-Vegetarian
cooking_time	480 min	480 min
calories_per_100g	500	high
fat_per_100g	20g	high
carbs_per_100g	50g	high
protein_per_100g	30g	high
allergenic	Gluten	Gluten

Table A.5.: Recipe Domain: Example user-item pair where item.allergenic = user.allergenic = Gluten