Evaluation of Automated Error Analysis with LLMs on three tasks: Schema-Matching, Sentiment Analysis and Price Prediction

Bachelor Thesis

presented by
Tim Gutberlet
Matriculation Number 1738709

submitted to the
Data and Web Science Group
Prof. Dr. Christian Bizer
University of Mannheim

April 2024

Abstract: The emergence of Large Language Models (LLMs) marks a significant milestone in the field of artificial intelligence. At least since the release of ChatGPT, these models have revolutionized the way people approach various tasks. This bachelor thesis investigates the effectiveness of LLMs, specifically GPT-3.5 and GPT-4, and traditional machine learning models in solving three distinct tasks: Schema-Matching, Sentiment Analysis, and Price Prediction. The results are then used to conduct automated error analysis, also using LLMs. The research aims to understand the capabilities of LLMs in identifying and classifying errors made by both the models themselves and traditional machine learning models. The theoretical section of the thesis presents a comprehensive overview of LLMs, tracing their development from statistical language models to neural networks, and the emergence of transformer-based architectures such as BERT and GPT. It also explores the concept of Prompt Engineering, which involves optimizing inputs to elicit desired outputs from LLMs, and Error Analysis, a process for comprehending model limitations and guiding improvements. The thesis concludes that in the Schema Matching task, the combination of GPT-4-p with Dynamic Few-Shot prompts demonstrates the best performance (F1 0.67) when the prompt includes all columns. This outperforms zero-shot prompts, particularly when only a single column is provided (F1 0.22 for GPT-4) and logistic regression models (F1 0.28). In Sentiment Analysis, both single-term and multi-term prompts were effective. Multi-term prompts (with an average F1 score of 0.64) slightly outperformed single-term prompts (with an average F1 score of 0.55) in most runs. No pattern of performance difference between zero-shot and few-shot prompts was noticeable. For Price Prediction, dynamic few-shot prompts with GPT-4-p achieved remarkable accuracy with an RMSE of 113508, outperforming traditional regression models. The best performing model (Random Forest) has an RMSE of 122178. In both Schema Matching and Price Prediction, GPT-4 demonstrated a high level of understanding of errors, categorising them into distinct classes and providing valuable insights. For Sentiment Analysis, the LLM also provided clear error classes, but they were not as insightful or exhaustive as those for the other tasks. The thesis concludes that LLMs seem to be able to effectively solve a range of tasks, especially when using optimized prompting strategies, and seem effective in conducting automated error analysis. However, they still face challenges in certain settings where they only provide obvious and general error classes or miss assigning errors in their answers. Future research could investigate larger datasets, different LLMs, and more complex tasks to gain a deeper understanding of the capabilities and limitations of LLMs in automated error analysis. This thesis contributes to the understanding of the capabilities of LLMs in automated error analysis and their application across different tasks, providing insights into LLM performance and model errors.

Contents

1 Introduction **1**
1.1 Introduction 1
1.2 Research Questions 2
1.3 Thesis Structure 3

2 Theoretical Framework **5**
2.1 Large Language Models 5
2.2 Prompt Engineering 9
2.3 Error Analysis 11
2.4 Schema Matching 12
2.5 Sentiment Analysis 14
2.6 Regression 16

3 Methodology **18**
3.1 Overview 18
3.2 Datasets 19
3.3 Preprocessing 20
3.4 Parameter Settings and Models 20
3.5 Baseline Analysis Prompt Design and Response-Mapping 23
3.6 Baseline Analysis ML Solutions 25
3.7 Evaluation Metrics 26
3.8 Error Analysis 29

4 Experimental Evaluation **34**
4.1 Schema Matching 34
4.1.1 Baseline Analysis 34
4.1.2 Error Analysis 38
4.2 Sentiment Analysis 41
4.2.1 Baseline Analysis 42

4.2.2 Error Analysis 45
4.3 Price Prediction 49
4.3.1 Baseline Analysis 49
4.3.2 Error Analysis 53
4.4 Results and Comparison 56

5 Conclusion 58
5.1 Summary 58
5.2 Future Work 59

A Further Experimental Results 70
A.1 Schema Matching 70
A.2 Sentiment Analysis 77
A.3 Price Prediction 90

B Program Code / Resources 104

C Prompts and Preprocessing 105
C.1 Preprocessing 105
C.2 Schema Matching 105
C.3 Sentiment Analysis 108
C.4 Price Prediction 112

List of Figures

3.1 Schema Matching Prompt Example Table 23
3.2 Confusion Matrix 26

4.1 ROC Curve Schema Matching 39
4.2 ROC Curve Dynamic-Few-Shot run Sentiment Analysis 45

A.1 ROC Zero-Shot run Sentiment Analysis 79
A.2 ROC Curve Fix-Few-Shot run Sentiment Analysis 79

List of Tables

3.1 Overview over performed experiments for each task 18

4.1 T2D-SM NH dataset statistics 34
4.2 Baseline analysis summary Schema Matching 35
4.3 MAMS ATSA dataset statistics 41
4.4 Baseline summary of weighted averages (Wei.) Sentiment Analysis 42
4.5 Error Class Overview for LLM runs (...) for Sentiment Analysis . 47
4.6 Error Class Overview for Log. Reg. (...) for Sentiment Analysis . 47
4.7 Vehicle dataset statistics 49
4.8 Baseline Summary Price Prediction 49
4.9 Confidence Analysis Price Prediction 53
4.10 Structured Explanation Overview - Fix-Few-Shot - Price Prediction 54
4.11 Summary of Results Used for Error Analysis - Price Prediction . . 56

A.1 Error Class Overview with # of occurrences - Schema Matching . 76
A.2 Baseline summary (...) - Only Single Term - Sentiment Analysis . 77
A.3 Baseline summary (...)- Multi Term & ML - Sentiment Analysis . 78
A.4 Structured Explanation Overview - Zero-Shot - Price Prediction . 90
A.5 Structured Explanation Overview - Fix-Few-Shot - Price Prediction 91
A.6 Structured Explanation (...) - Dynamic-Few-Shot - Price Prediction 91
A.7 Error Class Overview with # of occurrences - Structured Explanations 102
A.8 Error Class (...) - Unstructured Explanations - Price Prediction . . 103
A.9 Error Class Overview with # of occurrences - Lasso Regression . . 103

Chapter 1

Introduction

1.1 Introduction

At least Since the release of ChatGPT or LLaMa, Large Language Models (LLMs) have demonstrated their significance and power in solving language tasks [85]. Furthermore, LLMs have demonstrated proficiency in various new tasks, such as entity matching, that previously required significant amounts of fine-tuning [63]. To further increase the performance of LLMs, optimizing the instructions and examples given to the model, called "prompt engineering" has emerged as a new field of research [19]. Basic Prompt engineering methods such as few-shot prompting, where a few examples are provided within the prompt, have been shown to potentially increase performance in comparison to zero-shot prompting, where no examples are provided. [17]. To enhance the effectiveness of Zero-Shot and Few-Shot Prompting, it is essential to comprehend the nature and causes of errors the LLM does [48]. Moreover, recursively using LLMs to optimize prompts has been shown to be an effective way to increase performance [9]. A more advanced method of developing and adapting prompts for a particular domain is called prompt breeding and has been shown to outperform state-of-the-art prompting methods such as Chain-Of-Thought prompting [25]. Such methods could possibly be improved if LLMs would be able to analyse their arising error patterns and use these insights in the self-referential process. To gain a more profound understanding of arising error patterns, the occurring errors can be clustered into distinct error classes, called error analysis in the following [49]. However, manually understanding and clustering these error classes can be an inefficient and a time and resource-intensive process [18]. This thesis will investigate to what extent and how well LLMs can conduct error analysis in an automated way on errors done through zero-shot and few-shot prompting on the one hand, but also through solving tasks using a traditional machine learning model. This thesis also builds on the work of Burkhardt, who has already performed and evaluated automated error analysis for the task of entity matching [18]. This thesis will evaluate these techniques to three new task types: Schema-Matching, Sentiment Analysis and Price Prediction to test if and to what extent error analysis can be used to better understand and potentially improve the performance of solving these tasks with LLMs. A schema is a structured framework, typically hierarchical, that outlines how data is organised within a database, and schema matching is the process of discovering semantic matches between schema elements [56]. Typically, schema matching is executed manually by domain experts. This approach, however, is notably labor-intensive and time-consuming, and there have been developed multiple automated approaches to the schema matching task [23, 41]. Recent benchmarks have shown, that state-of-the-art LLMs such as GPT-4 can significantly outperform older approaches and models [21]. Due to the task's complexity and wide range of potential errors, it poses an ideal candidate for the error analysis. Sentiment Analysis involves the computational study of opinions, sentiments, emotions, and attitudes expressed in a text and aims to determine its polarity [42]. Aspect-Based Sentiment Analysis (ABSA) is an advanced branch of sentiment analysis that focuses on identifying the sentiment towards specific aspects within a text, rather than giving an overall sentiment score to the entire text [99]. LLMs have been shown to perform well on classical Sentiment Analysis, but they make enough errors on ABSA to perform error analysis [98]. The task of price prediction, for example for the price of used cars, has traditionally been solved by regression models [37]. However, LLMs have also been shown to be able so solve some basic optimization tasks to a satisfactory level [95]. It poses an interesting question, how well LLMs will be able to analyse errors done by a traditional regression machine learning model and if LLMs will be able to solve the price prediction task itself to a satisfactory level.

1.2 Research Questions

This thesis investigates the use of Large Language Models (LLMs) for automatic error analysis in three new domains: Schema Matching, Sentiment Analysis, and Price Prediction. The thesis seeks to answer the following questions:

1. What is the effectiveness of LLMs in performing the specified tasks when different prompting strategies are used, and how does their performance compare to that of conventional machine learning models in executing these tasks?

2. To what extent can LLMs be used to automatically analyse errors produced

by both themselves and conventional machine learning models?

The use of Large Language Models (LLMs) for automatic error analysis has the potential to enhance prompt quality and model performance [49]. Furthermore, it could offer a more profound comprehension of the structured responses produced by LLMs.

1.3 Thesis Structure

Theoretical Framework

The Theoretical Framework chapter establishes the basis for the entire thesis, concentrating on LLMs and their application in various tasks. It defines important terms and concepts that are relevant to the study, ensuring that the reader has a clear understanding of the topics discussed throughout the thesis. It then explores the historical development of LLMs, highlighting significant milestones and discusses state-of-the-art models. Due to the significance of prompt engineering and error analysis in the application of LLMs, this thesis explores these sub-fields in detail, providing insight into their roles and methodologies in separate chapters. Additionally, the chapter examines the tasks Schema Matching, Sentiment Analysis, and Linear Regression. It provides historical context, current methods, and state-of-the-art approaches for each task, with a focus on how LLMs are employed to solve them. This chapter establishes the foundation for the research by presenting a theoretical basis, justifying the research questions, and outlining the significance of the experiments.

Methodology

The methodology chapter describes the research design, data collection methods, and analytical techniques used to conduct the experiments. This chapter is important because it ensures that the results are reproducible and because it explains how the research was conducted. The chapter describes the experimental setup, including the creation of prompts for zero-shot and few-shot learning, and the setup for solving each taks with a classical machine learning model. In addition, the error analysis approach is explained, highlighting how errors are defined, classified and analysed to provide deeper insights into the capabilities and potential of the LLMs. This chapter details each step of the experimental process to ensure the validity and reliability of the results.

Experimental Evaluation

In the Experimental Evaluation chapter, the thesis presents a detailed analysis of the experiments conducted for the Schema Matching, Sentiment Analysis and Price Prediction tasks. The results are divided into baseline and error analyses for each task type. The baseline analysis compares the performance of LLMs in zero-shot and few-shot scenarios with that of traditional machine learning models, providing a comprehensive overview of their effectiveness. The Error Analysis section provides an in-depth examination of the errors made by LLMs. Through quantitative and qualitative assessments, this chapter examines the strengths and limitations of performing error analysis with LLMs for the specified tasks. The concluding discussion summarizes and compares the findings from all three tasks.

Conclusion

The conclusion concisely summarizes all results and findings of thesis and addresses the research questions. It also outlines potential areas for future work.

Chapter 2

Theoretical Framework

2.1 Large Language Models

Natural Language Processing (NLP) is a field of artificial intelligence that concentrates on creating and implementing systems and algorithms for human language interactions [43]. *Language Modelling* is a fundamental part of the field of NLP, with the goal of predicting the probability of a future or missing sequence of words occurring in a sentence[100]. For instance, if presented with an incomplete sentence like 'He closed his laptop and...', the model can predict the most probable next word or phrase, such as 'went home' or 'got a coffee'. Large language models (LLMs) use extensive text data to learn intricate language patterns, allowing them to perform various NLP tasks with exceptional proficiency, sometimes surpassing human performance [60]. Tasks can vary from simple translation or summarisation to question answering and creative writing. To comprehend the evolution of LLMs, it is crucial to understand the historical progression of NLP from statistical to neural language and then from pre-trained language models (PLMs) to LLMs [60].

History and Origins of Statistical Language Models

The development of *Statistical Language Models (SLMs)* was initially driven by the need to improve machine translation and speech recognition systems. One of the earliest and most influential models was the n-gram model [28]. An n-gram is a series of exactly n tokens or words, and a n-gram model predicts the probability of a word based on the occurrence of the preceding n-1 words. This approach treats language as a Markov process. A Markov process is a stochastic process in which the probability of transitioning to any future state depends solely on the current state and not on the sequence of events that preceded it [27]. SLMs have been extensively used in information retrieval [13] and natural language processing [12, 100]. However, they also have serious limitations. Specifically, they are unable to capture long-range dependencies and suffer from the curse of dimensionality. As n increases, the feature space (i.e. the number of unique n-grams) grows exponentially.[20, 75, 100].

Neural Networks in Language Modelling

Neural networks (NN) are computational models inspired by the structure and function of the human brain [6]. They consist of layers of interconnected nodes or neurons, where each connection can transmit a signal from one node to another [77]. The nodes in each layer perform simple operations on the signals they receive, and the strength of the connections between the nodes (the weights) is adjusted during the training process.

Neural language models (NLMs) are a subset of neural networks specifically designed to understand, generate, and manipulate human language [15]. They are trained on large corpora of text data and learn the statistical properties of the language, including grammar, syntax, and semantics. This allows for the prediction of the following word in a sentence, the generation of coherent text, language translation, and other language-related tasks.

Multilayer Perceptrons (MLPs) are a simple form of neural networks. They consist of an input layer, one or more hidden layers, and an output layer [71]. Each layer is made up of neurons fully connected to the neurons in the previous and following layers. MLPs use a feedforward architecture, which means that data moves only in one direction from input to output, with no loops in the network. Backpropagation is a technique used to adjust the weights of connections based on the error between predicted and actual outputs [67]. Multi-Layer Perceptrons (MLPs) are capable of handling various data-driven tasks, including classification and regression [58]. However, their effectiveness in neural language models is limited due to their feedforward and fixed-size input nature [67], which hinders their ability to effectively process sequences or the context of words in a sentence.

Recurrent Neural Networks (RNNs) are designed specifically to address the limitations of previous methods in processing sequences and contextual information [55]. Unlike MLPs, RNNs have loops in their architecture, allowing information to persist. In an RNN, each neuron can transmit information not only forward to the next layer but also backward to itself across different time steps [40]. Recurrent Neural Networks (RNNs) can maintain memory by considering both current and previous inputs. They have been successfully used for tasks such as text generation, machine translation, and speech recognition [78]. However, RNNs may encounter difficulties with long-term dependencies due to problems such as vanishing or exploding gradients during training. To tackle these problems, more sophisticated versions of RNNs have been created, such as Long Short-Term Memory (LSTM) networks [78, 30]. By incorporating a unique structure that includes memory cells and gates (input, output, and forget gates), Long Short-Term Memory networks deal with the vanishing gradient problem [88]. This capability enables them to perform remarkably well in tasks that involve sequential data, such as speech recognition and text generation, where traditional neural networks falter due to their inability to maintain long-term dependencies. [30].

Moreover, *word2vec*, a group of more shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words[1] have emerged and have shown to be very effective in NLP tasks [54, 59]. Such developments have had a significant impact on NLP as they led to a more widespreach use of language models beyond word simple word sequence modeling [100].

Pre-Trained Language Models (PLMs)

Still, models such as word2vec were limited by their inability to capture the full complexity of language, including syntax and long-range dependencies. The introduction of transformer-based architectures, notably with the publication of the ”Attention is All You Need” paper [89] marked a significant milestone [46]. This architecture used the self-attention mechanism, allowing the model to weigh the significance of different words in a sentence regardless of their positional distance. This original transformer is a sequence-to-sequence model [82] and consists of an encoder and a decoder. For example, in machine translation, a transformer model can translate a sentence from English to German, taking into account the context of each word for a more accurate translation. The encoder transforms the input sequence (e.g. a sentence in the source language) into a set of representations, enriching them with context [3]. It achieves this through layers that each consist of two main components: a multi-head self-attention mechanism and a feedforward network [89]. A *multi-headed attention mechanism* means that the transformer uses several attention mechanisms (heads) in parallel. The self-attention mechanism allows the encoder to weigh the significance of different parts of the input relative to each other, capturing the intricacies of the sequence. The decoder generates the output sequence (e.g., the translated sentence in the target language) from the encoder's representations. It mirrors the encoder's structure but adds a third component: a multi-head attention mechanism that links the decoder and encoder, enabling the decoder to focus on relevant parts of the input sequence. The decoder's self-attention mechanism is masked to ensure predictions are based only

[1] https://en.wikipedia.org/wiki/Word2vec

on previously known outputs, facilitating sequential generation [89]. Following the groundbreaking introduction of the transformer architecture, several influential models have emerged. The *Bidirectional Encoder Representations from Transformers* introduced, commonly known as the BERT, is a form of Transformer architecture that aims to pre-train deep bidirectional representations from unlabelled text [22, 81]. It does this by simultaneously conditioning on both the left and right contexts at every layer. This approach allows the model to gain a more nuanced understanding of language, enabling it to perform better than previous models on various natural language processing tasks [11]. A PLM such as BERT follows a semi-supervised approach, where the first stage involves pre-training on a large set of unlabeled training data giving it an understanding of the general structure of the content without being redirected to any specific task. The second stage involves fine-tuning, where the model is further trained on a smaller, task-specific dataset, enabling it to excel in specific tasks [22]. *Generative Pre-trained Transformer (GPT)* extends the transformer model into generative tasks [69]. Unlike BERT, which focuses on bidirectional context, GPT utilizes a left-to-right approach, enabling the generation of coherent and contextually relevant text. Since the introduction of the original GPT, there have been several iterations each significantly increasing in size and complexity [102]. These newer versions have shown remarkable improvements in generating text that is increasingly indistinguishable from that written by humans, demonstrating advanced understanding of context, subtlety, and even humor [17, 102].

Current State

PLMs have been observed to exhibit unexpected abilities, commonly known as 'emergent abilities', during their evolution [91]. These abilities are not explicitly programmed into the models but arise as the models scale in size and complexity [39]. For example, large PLMs have demonstrated the ability to solve problems, reason, and comprehend abstract concepts at levels beyond their creators' expectations [92]. They can perform tasks such as arithmetic calculations and generate software code from descriptions with increasing accuracy as the models grow larger and are often referred to as *Large Language Models* [32]. Currently, there are several LLMs available, some of which are open-source while others offer their services through paid APIs. Relevant open-source models include LLama2[2] [85], Bloom[3] [93], and Falcon[4] [7], which have up to 70 billion, 176 billion, and 180

[2] https://ai.meta.com/llama/

[3] https://bigscience.huggingface.co/blog/bloom

[4] https://falconllm.tii.ae/

billion parameters, respectively. The most popular paid models are OpenAI's[5] GPT-3.5 and the newer, more expensive, and better-performing version, GPT-4, which gained popularity through the ChatGPT[6] platform, making LLMs accessible to a wide audience [96, 62]. Other models competing at the level of GPT-4 include Google's Gemini[7] [83] and Anthropic's Claude3[8].

2.2 Prompt Engineering

Fundamentals of Prompt Engineering

Prompt Engineering for LLMs is a significant area of study that concentrates on creating and optimizing inputs, known as prompts, to elicit desired outputs from these systems [19]. In this context, a prompt is essentially an instruction or query given to an LLM, designed to guide the model towards generating specific information, answers, or content. For instance, a basic prompt such as 'Provide information on the Renaissance' yields general information. The prompt 'Provide a detailed analysis of the Renaissance, focusing on its origins, key figures, and impact on art, science, and society' is a more refined version that directs the LLM to a specific and detailed response. Prompt engineering is crucial as it can significantly affect the accuracy, relevance, and quality of the model's output [38]. To achieve effective prompt engineering, it is crucial to have a comprehensive understanding of the model's capabilities and limitations. Additionally, creativity and strategic thinking are necessary when communicating with the LLM to fully leverage its potential [50]. Mastering prompt engineering is essential for users who want to utilize the full power of LLMs [19].

Foundational Strategies for Prompt Engineering

Zero-shot prompting refers to the scenario where an LLM is asked to perform a task without any prior examples. The model relies solely on its pre-existing knowledge and training to generate a response [17]. The key to successful zero-shot prompting lies in crafting prompts that are clear, precise, and self-contained, ensuring the model has all necessary information to understand and fulfil the request accurately.

Role prompting is an innovative approach where the prompt specifies a particular role for the LLM to assume, such as a tutor, a critic, or a storyteller [101].

[5] https://openai.com/
[6] https://chat.openai.com/
[7] https://gemini.google.com/app
[8] https://www.anthropic.com/claude

This strategy guides the model's tone, style, and type of response, aligning it more closely with the user's expectations [19].

The "be clear and precise" strategy emphasizes the importance of crafting prompts that are unambiguous and detailed. By explicitly stating the desired output format, level of detail, and any specific considerations, users can significantly improve the relevance and quality of the responses generated by the LLM [19].

One-shot and few-shot prompting involve providing the LLM with one or a small number of examples, respectively, to demonstrate the desired task or response format [10]. These examples serve as a template for the model, enabling it to better understand and replicate the requested output style or content structure in its response [19]. Few-Shot Prompting has been shown to significantly improve LLM performance for various tasks [51]. *Fixed Few-Shot Prompting* is a static version of few-shot prompting that provides a predefined set of examples to a model. *Dynamic Few-Shot Prompting* introduces a flexible and adaptive approach. The method employs algorithms, such as the k-nearest neighbours (KNN) algorithm, to dynamically find and utilize the most similar samples to the task at hand from a training set dataset. This means that the set of examples used for prompting can change based on the specific requirements of each new task or query presented to the model, and therefore potentially improve performance [61]. The KNN algorithm classifies a new data point on the basis of the majority vote of its 'k' nearest neighbours in the feature space [29]. The value for a new point is predicted by averaging the values of its 'k' nearest neighbours in the regression task. To process text data, it is converted into a numerical format.

Temperature controls the randomness in the LLM's response generation process. [33] A lower temperature results in more deterministic and confident responses, while a higher temperature encourages creativity and variability [72]. Adjusting the temperature can be useful when seeking to balance between the predictability and novelty of the generated content. As LLMs are non-deterministic, trying a prompt several times can be beneficial to get the best result.

Advanced Strategies for Prompt Engineering

Chain of thought prompting (CoT) encourages LLMs to break down complex problems into smaller, more manageable steps before arriving at a conclusion [92]. This technique is particularly useful for tasks requiring logical reasoning or complex problem-solving . By structuring the prompt to guide the LLM through a step-by-step reasoning process, users can obtain more accurate and interpretable answers [94].

Prompt breeding involves iteratively refining prompts based on the performance of previous iterations [25]. This method treats prompt creation as an evolutionary process, where each generation of prompts is adjusted based on feedback from the LLM's responses . Over time, this leads to highly optimized prompts that can elicit better performance from the model on specific tasks even outperforming advanced prompting methods such as CoT [25].

2.3 Error Analysis

Introduction to Error Analysis

The use of LLMs to automatically analyse errors that occur after solving specific tasks with LLMs has been shown to be a potentially effective method for improving performance and better understanding the structure of errors that occur in language translation [48]. In 2022, Lu et al. took the first step towards automatic error analysis, which could potentially enable LLMs to achieve human-like language translation scores by grouping errors into task-specific error classes [48]. In 2024 Lu et al. proposed a new method called error analysis prompting, which achieved promising results and showed that GPT-4 could produce human-like evaluation of machine translations [49]. This paper called "Toward Human-Like Evaluation for Natural Language Generation with Error Analysis" presents an innovative method for evaluating machine translation (MT) quality with LLMs. The motivation behind this work stems from the observation that while LLMs have shown state-of-the-art performance in assessing MT quality at a system level, their performance at a segment level (i.e., individual sentences or smaller text units) has been less impressive, especially in terms of providing detailed, interpretable feedback on translation errors. To address this, the authors propose a new method called Error Analysis Prompting (EAPrompt), which integrates two approaches: CoT and Error Analysis (EA). The CoT approach encourages LLMs to "think aloud" by breaking down their thought process into intermediate steps before arriving at a conclusion. EA, on the other hand, involves the identification and categorization of errors in a translation, distinguishing between major errors (which significantly impact the translation's accuracy or grammaticality) and minor errors (which are less severe and might relate to stylistic preferences or minor imperfections). The EAPrompt method is designed to mimic the human evaluation framework known as Multidimensional Quality Metrics (MQM). This approach enables LLMs to produce explainable and reliable evaluations of MT quality, identifying specific errors and scoring translations based on the severity of these errors. The authors conducted experiments using datasets from the WMT22 metrics shared task and found that EAPrompt significantly improves LLMs' performance in evaluating MT quality at both system and segment levels compared to existing methods [49]. Error analysis in this context refers to the systematic identification and categorization of errors in a LLM response. The capability of error analysis lies in its potential to provide detailed feedback on the types and severities of errors present in a task, thereby enabling more nuanced and human-like evaluations of response quality. This method allows for the assessment of responses in a way that aligns more closely with human judgment, particularly by distinguishing between error classes and evaluating their impact on the overall quality of the response [48]. This new area of research therefore poses the interesting questions if LLMs are capable of automatic human like evaluation of errors in tasks beyond MT.

Black-Box vs. White-Box

In machine learning, the terms black-box and white-box models refer to the extent to which the internal workings of a model are understandable and interpretable by humans.

Black-box models are models whose internal workings are not easily interpretable or understandable by humans [47]. These models can make predictions or decisions without providing a clear explanation or insight into how they arrived at these conclusions. While users can see the input and output, they cannot see the process in between.

Examples of black-box models include LLMs and complex ensemble methods. These models are renowned for their high accuracy and predictive power in various tasks. However, they are often criticised for their lack of transparency, which makes it challenging to understand the reasoning behind certain decisions or predictions [47].

White-box models are machine learning models whose internal workings are fully understandable and explainable. These models allow users to see and understand the decision-making process, including how input data is transformed into output predictions or decisions [47]. Examples of white-box models include linear regression, decision trees, and logistic regression, which will be explained later in the chapter. These models might not always match the predictive performance of more complex black-box models, but they offer the advantage of interpretability, making them valuable in scenarios where understanding the decision process is critical, such as in medical diagnostics, financial modeling, and legal applications.

2.4 Schema Matching

Introduction to Schema Matching

Schema Matching is the process of identifying correspondences between elements in two or more schemas [23, 45]. A schema, in this context, refers to the structural definition of data, which may include databases, XML files, or other structured data formats. [52] The goal of schema matching is to enable interoperability and data sharing among different systems by discovering semantic correspondences between elements of the different schemas. For example, consider the databases of two online bookstores: Store A has a Books table with columns BookID, Title, AuthorName, and Price. Store B uses an Inventory table with ISBN, BookTitle, Author, and SalePrice. Schema matching would establish that BookID matches ISBN, Title corresponds to BookTitle, AuthorName to Author, and Price to SalePrice. This matching process can be carried out manually, semi-automatically, or automatically, with varying degrees of human intervention and computational assistance. The complexity of schema matching stems from the heterogeneity of schema representations, naming conventions, and data semantics [56]. As a result, the field has seen the development of a multitude of approaches and algorithms, each aiming to address specific aspects of the matching challenge. Schema matching methods can be grouped by the information they utilize for finding matching element [70, 21]: Label-Based Methods utilize textual similarity of schema element names, such as table and column names, to identify matches. Instance-Based Methods analyze actual data within elements, looking for common patterns or values to infer matches. Structure-Based Methods leverage relationships and arrangements within schemas, such as entity relationships, to find matches based on the structural context. Hybrid methods combine all previous methods to leverage their strengths and mitigate individual weaknesses [21].

Review and History of Schema Matching Systems

The development of schema matching systems has evolved significantly since the late 1990s and early 2000s. Initially, schema matching was predominantly manual or relied on simple automated methods that focused on names and data types of schema elements. In 2001, Rahm and Bernstein's work in 'A survey of approaches to automatic schema matching' categorised schema matching techniques into categories such as schema-level and instance-level approaches, providing a framework for future innovations [70]. With the growth of the internet and semantic web, schema matching became more complex, leading researchers to explore more nuanced approaches. Doan et al.'s (2002) work, 'Learning to Match Ontologies on the Semantic Web', represents a shift towards improving matching accuracy through machine learning [24]. Other notable systems for automatic schema matching from around that time are COMA [23], SEMINT [45], or Cupid [52].

Newer techniques started to address not only structural and syntactic variations but also the semantic context of the data being matched. Recent advancements have led to the integration of context-aware and ontology-based methods. This is exemplified by works such as Shvaiko and Euzenat's 'Ontology matching: state of the art and future challenges' (2013), which explores the role of semantic technologies in improving match quality [80]. In addition, the emergence of deep learning has brought new techniques, including the utilization of neural networks for feature extraction and similarity computation, with the goal of capturing more profound semantic connections between schema elements. An example worth mentioning is the use of Graph Neural Networks (GNNs) for schema matching, as discussed in Li et al.'s 'Deep Graph Matching Consensus' (2020) [26]. This study demonstrates how deep learning techniques can effectively address the complexity of matching tasks by learning from graph structures that represent schemas.

Recently, LLMs have emerged as a state-of-the-art method for solving schema matching tasks [21]. Sheetrit et al. (2024) presents an innovative approach named ReMatch [79]. This approach leverages the capabilities of LLMs to significantly improve the efficiency and accuracy of schema matching without requiring predefined mappings, model training, or access to source schema data. This method creates structured document representations for both source schema attributes and target schema tables. It employs semantic relevance for document retrieval and utilizes LLMs to generate a ranked list of potential matches. The experimental results on large real-world schemas highlight the effectiveness of ReMatch, demonstrating its superiority over traditional machine learning approaches in schema matching tasks. A recent benchmarking by Der and Bizer has also shown GPT-4 to perform very well on instance-based and label-based schema matching [21].

2.5 Sentiment Analysis

Introduction to Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a field of study in NLP and computational linguistics [98]. Its aim is the systematic identification, extraction, quantification and study of affective states and subjective information. This discipline primarily focuses on analyzing opinions, sentiments, evaluations, attitudes, and emotions from written language [68]. The field of sentiment analysis has a broad range of applications, including market research, consumer behavior analysis, product reviews, social media monitoring, and customer feedback [14, 99].

A more nuanced approach to sentiment analysis is Aspect-Based Sentiment Analysis (ABSA), which not only identifies the sentiment expressed in a text but also the specific aspects or features that the sentiment is directed towards [99]. ABSA can be divided into two sub-tasks: aspect term sentiment analysis and aspect category sentiment analysis [36].

Aspect Term Sentiment Analysis aims to extract aspect terms (specific entities or attributes being discussed) from the text and determine the sentiment expressed towards each of them [36]. For example, in the review, 'The phone's battery life is impressive, but the camera is disappointing,' the aspect terms would be 'battery life' with a positive sentiment and 'camera' with a negative sentiment.

Aspect Category Sentiment Analysis involves identifying broader categories or domains to which these aspects belong and assessing the sentiment towards them [36]. In the given example, the categories of aspects are 'battery' and 'camera', each assigned a sentiment based on the overall sentiment expressed towards the aspects falling under them.

Review and History of Sentiment Analysis Systems

The origins of automated classification in sentiment analysis can be traced back to the paper by Turney titled "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews" [87]. This seminal work laid the foundation for subsequent research in sentiment analysis by introducing a simple yet effective method for classifying the semantic orientation of textual content, marking a significant departure from earlier, more manual and lexicon-based approaches. Over the years, the field has witnessed rapid advancements, particularly with the advent of machine learning and deep learning techniques, which have significantly improved the accuracy and efficiency of sentiment analysis systems [97]. Notable contributions include the development of supervised learning models, such as support vector machines (SVM) and deep neural networks, which have demonstrated remarkable proficiency in understanding context and nuances in language [57, 90].

The investigation into the use of Large Language Models (LLMs) for sentiment analysis reveals that while LLMs excel in simple sentiment classification tasks, their performance declines in more complex areas requiring detailed sentiment understanding or structured sentiment information, for example in areas such as ABSA [98]. The study conducted by Zhang et. al [98] compares LLMs to smaller, specialized models across a variety of tasks, finding that LLMs are particularly effective in few-shot learning scenarios where data is limited. However, the research also highlights the limitations of current evaluation methods for assessing LLMs' capabilities in sentiment analysis, suggesting the need for a more comprehensive and realistic benchmark, named SENTIEVAL. This work underscores the potential and challenges of employing LLMs in sentiment analysis, pointing towards the necessity for further advancements to fully leverage their capabilities in nuanced sentiment understanding.

2.6 Regression

Introduction to Regression

Regression analysis constitutes a fundamental statistical approach used to examine the relationship between a dependent variable and one or more independent variables [74]. The primary aim of this analysis is to model the expected value of the dependent variable based on the provided values of the independent variables. In this context, the dependent variable is the interesting result, influenced by the predictors or independent variables. Regression analysis is instrumental in forecasting, time series modelling, and inferring causal relationships between variables [53]. It enables the identification of significant predictors among the independent variables and quantifies the strength of their impact on the dependent variable. The coefficients of regression, which represent the difference in the dependent variable for a single unit change in an independent variable, having all other variables remain the same, are key to interpreting these relationships.

Review of Regression Systems

The historical development of regression systems traces back to the 19th century, initially emerging from the field of statistics [53]. Over time, various regression methods have been developed to cater to different types of data and research questions.

Linear regression [34], one of the oldest forms, is a statistical approach that assesses the correspondence between a dependent variable and one or more independent variables by fitting a linear equation to the provided data. The most common method is simple linear regression, that functions with one independent variable [53]. The equation for a simple linear regression line is

$$y = \beta_0 + \beta_1 x + \epsilon,$$

where y represents the dependent variable, x is the independent variable, β_0 is the y-intercept, β_1 is the slope of the line, and ϵ is the error term that captures the deviation of the observed values from the line. Multivariable or multiple linear regression extends this concept to include two or more independent variables [76], leading to

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon.$$

The primary objective of linear regression analysis is to find the values of the coefficient(s) (β) that minimize the sum of the squared differences between the observed and predicted values, a method known as Ordinary Least Squares (OLS).

Lasso Regression (Least Absolute Shrinkage and Selection Operator) [84] is a type of linear regression that adds a penalty equivalent to the absolute value of the magnitude of coefficients. This penalty term is represented as $\lambda \sum |\beta_i|$, where λ is a tuning parameter. Lasso regression can lead to some coefficients being shrunk to zero, effectively performing variable selection and leading to simpler, more interpretable models.

Ridge Regression [31] applies a penalty term to the linear regression model, but unlike Lasso, the penalty is the square of the magnitude of the coefficients: $\lambda \sum \beta_i^2$. This method is particularly useful when dealing with multicollinearity or when the number of predictors exceeds the number of observations. Ridge regression shrinks the coefficients and helps to reduce model complexity and multicollinearity.

Elastic Net Regression [103] is a hybrid of Lasso and Ridge regression methods. It includes both penalties: $\lambda_1 \sum |\beta_i| + \lambda_2 \sum \beta_i^2$. Elastic Net aims to combine the best of both worlds, maintaining variable selection properties of Lasso while also allowing the model to retain the grouping effect of Ridge regression.

Logistic Regression, specifically, is a type of regression analysis used for predicting the result of a categorical dependent variable based on one or more independent variables [64]. The model predicts the probability that the dependent variable belongs to a particular category. For binary logistic regression, the model uses the logistic function to compute a binary dependent variable, with the equation given by

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n,$$

where p is the probability of the dependent variable equaling a 'success' or '1'. This method is widely used in various fields, such as in medical fields [35], in marketing [4] for predicting customer behavior, and in finance for credit scoring [5].

Other notable and more advanced methods that can be used to solve regression tasks are: Decision Tree, Polynomial Features and Random Forest. A *decision tree* is a flowchart-like tree structure in which an inner node corresponds to an attribute, a branch refers to a decision rule, and each leaf node relates to the result, used for both classification and regression tasks [73]. *Random forest* is an ensemble learning method that functions by creating multiple decision trees at training time and returning the class that is the basis of the classes (classification) or mean prediction (regression) of each of the trees [16, 2].

Chapter 3

Methodology

3.1 Overview

The aim of the experiments is to assess the problem-solving abilities of an LLM in three tasks: Schema Matching, Sentiment Analysis, and Price Prediction. The LLM's performance will be compared to that of traditional Machine Learning Models (MLs). This first part will be called baseline analysis in the following. Additionally, the results will be presented to an LLM, along with an explanation of the outcomes, and the LLM will be asked to identify and cluster error classes. This part will be called error analysis in the following.

Table 3.1 shows a structural overview of the analysis that will be further explained in the Methodology section and evaluated in the Experimental Evaluation section for each task. For the Schema Matching task, only the unstructured explanations were created.

Prompts / Analysis	**Model**
Baseline Analysis:	
Zero-Shot, Fix Few-Shot, Dynamic Few-Shot	GPT-3.5 & GPT-4
Traditional ML	Regression model
Error Analysis:	
Confidence Analysis	GPT-3.5
Structured & Unstructured Explanations	GPT-4
Error Classes	GPT-4

Table 3.1: Overview over performed experiments for each task

3.2 Datasets

For the study experiments, three different datasets, one for each task, will be used. For these tasks' test sets, a downsampling technique was used to reduce dataset size by randomly selecting a subset of the data. This approach helps manage large datasets by decreasing the volume of data to be processed while preserving the original dataset's statistical characteristics. It was used in this context because handling the entire dataset is not feasible due to financial and computational limitations, and to simplify the subsequent analysis.

Schema Matching

The T2D-SM NH[1] Dataset is a component of the Web Data Commons Schema Matching Benchmark [21], focusing on instance-based schema matching methods. This particular dataset, without column headers, emphasizes matching Web tables[2] to DBpedia tables[3] based on instance data alone. It comprises 178 table pairs across 28 DBpedia classes (representing topic domains) with 445 positive and 2197 negative column correspondences. This variant tests purely instance-based schema matching capabilities, distinct from its counterpart T2D-SM-WH which includes column headers for a blend of instance- and label-based matching. The benchmarking for GPT-4 has determined an F1 value of 0.75, indicating that the LLM still makes enough errors for error analysis on this dataset. In comparison, the F1 value for the T2D-SM-WH dataset is 0.93.

Sentiment Analysis

The Multi Aspect Multi Sentiment (MAMS) aspect term sentiment analysis (ATSA) Dataset[4] was introduced by Jiang et al. [36] and will be used for the sentiment analysis task. This dataset was created to address the limitations of previous datasets. It includes sentences with at least two different aspects with differing sentiment polarities, making it a more challenging and realistic dataset for LLMs to analyze. This also leads to a sufficient number of errors for the following analysis. The dataset comprises 13,854 instances for the ATSA task and 8,879 instances for the aspect-category sentiment analysis (ACSA) task, making it significantly larger than the SemEval-2014 Restaurant Review dataset [66].

[1]https://webdatacommons.org/structureddata/smb/#toc5

[2]https://webdatacommons.org/webtables/

[3]https://www.dbpedia.org/

[4]https://github.com/siat-nlp/MAMS-for-ABSA/tree/master/data/MAMS-ATSA/raw

Price Prediction

The 'Vehicles' dataset[5] from Kaggle was used for the price prediction task. Specifically, the 'Car Details v3' dataset was selected due to its sufficient number of entries and a variety of numerical and categorical features. This dataset provides information on the features and brands of used cars, with 12 independent variables and 8128 entries. It has been downloaded 141,000 times on Kaggle and is therefore frequently used. The provided dataset contains used car prices in Indian Rupees (INR), spanning the years 1997 to 2016.

3.3 Preprocessing

The dataset preprocessing consisted of several steps. Firstly, the datasets underwent manual checks to identify and correct any obvious errors that could negatively impact the automated error analysis later. Next, the CSV and XML datasets were parsed into a Python dataframe representation. Then, the test set for each task was randomly sampled down due to time and cost constraints. To conduct dynamic few-shot analysis for each task, a KNN algorithm[6] was used to identify the most similar entries from the training set. To process text data, it was converted into a numerical format using a TF-IDF Vectorizer[7]. For the fix-few-shot analysis, samples from the training set were randomly sampled once. For the Schema matching task, table representations were created as string representations for the prompts. To limit the size of the prompts, 5 entries were sampled at random from each column.

3.4 Parameter Settings and Models

This thesis utilises the state-of-the-art GPT Models developed by OpenAI[8], which are among the most relevant and commonly used LLM models available. Additionally, the OpenAI paid API[9] was used to access the models, eliminating the need for own computing power. The OpenAI API is well documented and can be accessed

[5] https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho/data?select=Car+details+v3.csv
[6] https://scikit-learn.org/stable/modules/neighbors.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[8] https://platform.openai.com/docs/models
[9] https://openai.com/blog/openai-api

directly through HTTP[10], Python[11], Node.js[12] or LangChain[13] libraries. Python was used for this thesis. The Langfuse[14] library was used to monitor and log the API requests and responses. The cost of the models is based on per-token usage. It is generally estimated that 75 words equate to approximately 100 tokens[15].

To solve the Tasks with a machine learning model, the sklearn python library has been used[16]. Here, several parameters were potentially adjusted. The C[17] parameter controls the strength of regularization, with smaller values indicating stronger regularization and larger values indicating weaker regularization, thereby influencing the complexity and generalization capability of the model. The class_weight parameter allows for the adjustment of weights assigned to different classes, aimed at addressing imbalances by influencing the cost function to prioritize correctly predicting underrepresented classes more heavily.

The following paramters and GPT-3.5 and GPT-4 models have been used in the experiments:

Parameters

The OpenAI API has two required and several optional parameters. The 'model' parameter specifies the GPT Models to be used for the request. The 'messages' parameter includes the prompt and all previous conversation that the model should use to draft its response. The messages in the system can be categorised as user messages, system messages, assistant messages or tool messages and can affect the response of the LLM[18]. Optional parameters have also been used in some cases. For example, the 'max_tokens'[19] parameter was set to 15 for some of the baseline evaluation prompts to prevent the LLM from generating responses longer than one word. To ensure reproducible outcomes, all models used in the experiments had their temperature set to 0. All other parameters were kept at their default values.

[10] https://platform.openai.com/docs/api-reference
[11] https://github.com/openai/openai-python
[12] https://github.com/openai/openai-node
[13] https://python.langchain.com/docs/integrations/llms/openai
[14] https://langfuse.com/docs/sdk/python
[15] https://platform.openai.com/tokenizer
[16] https://scikit-learn.org/stable/
[17] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[18] https://platform.openai.com/docs/api-reference/messages/object
[19] https://platform.openai.com/docs/api-reference/chat

Models

GPT-3.5-turbo-0125

GPT-3.5-turbo-0125 is the newest model within GPT-3.5 set. It has been fine-tuned to improve accuracy when responding in requested formats and returns a maximum of 4,096 output tokens. It is also optimised for better dialogue results than previous GPT-3.5 models. The model has a context window of up to 16,385 tokens and costs $0.5 per 1M tokens for input and $1.5 per 1M tokens for output. The training data goes up to September 2021. When ”GPT-3.5-t” or ”GPT-3.4” is used below, it refers to the GPT-3.5-turbo-0125 model.

GPT-4-0613

GPT-4 represents a significant advance over its predecessor, GPT-3.5, in the evolution of the language models developed by OpenAI. The key differences lie in its increased size and complexity, enabling GPT-4 to deliver responses with greater accuracy and nuance. This is achieved through a larger dataset and a higher number of parameters, enabling the model to capture and process complex instructions more effectively. Significant improvements in GPT-4 include its enhanced ability to produce contextually relevant results, a reduction in biased or inappropriate content, and superior performance in specialised domains. These advances are attributed to refined training processes and security mechanisms, ensuring that GPT-4 can be used more responsibly and effectively in a wider range of applications[20]. GPT-4-0613 is the latest in the GPT-4 model series, with a context window of up to 8,192 tokens and training data that goes up to September 2021.

GPT-4-0125-preview

GPT-4-0125-preview, is the latest model of the GPT-4-turbo-preview set that is more accurate and has more features than previous GPT-4 models, while also being cheaper. This new version also aims to reduce the number of cases where the model doesn’t complete a task. It has a context window of 128,000 tokens and the training data goes up to December 2023. It costs $10 per 1M tokens for input and $30 per 1M tokens for output. Consequently, GPT-4-0125-preview is less expensive and performs better than GPT-4-0613. When ”GPT-4-p” or ”GPT-4” is used below, it refers to the GPT-4-0125-preview model.

[20] https://openai.com/blog/new-embedding-models-and-api-updates

```
|   Column A-0 | Column A-1    | Column A-2    |   Column A-3 | Column A-4          |
|-------------:|:--------------|:--------------|-------------:|:--------------------|
|        307   | SG            | 30.12 km      |         2579 | Plattenspitz        |
|        280.5 | GR            | 32.37 km      |         2264 | Crap Sogn Gion      |
|        270.1 | GR/UR         | 60.92 km      |         3256 | Gross Düssi Piz Git |
|        298.8 | GR/SG         | 27.29 km      |         3114 | Piz da Sterls       |
|        278.9 | GR            | 34.25 km      |         2412 | Crest da Tiarms     |
```

Figure 3.1: Schema Matching Prompt Example Table

3.5 Baseline Analysis Prompt Design and Response-Mapping

To assess the impact of prompting techniques and improve our understanding of LLMs error analysis capabilities, three different prompts for each task will be deployed: zero-shot, fix-few-shot, and dynamic-few-shot. The fix-few-shot and dynamic-few-shot prompts are identical and will be referred to as a few-shot prompt in the following. All prompt designs used for the experiments can be found in the appendix.

Schema Matching

The preprocessing stage has created table representations for the Web tables (Table A) and DBpedia (Table B), which will be used in this prompt. An example string representation of a table is shown in figure 3.1. The table layout has been taken from the Table-GPT paper [44] prompt[21] and used for the schema matching prompts. There are two types of instructions: single-column prompt and multi-column prompt. They will be explained in the following.

Single-Column Prompt

This prompt was created to evaluate how well the GPT models perform when asked to identify only whether a particular column pair matches or not. If this prompt works sufficiently well, it would make subsequent error analysis easier and more straightforward than an LLM response with pairs for all columns. It takes as input Table A and Table B and the column pair to be checked. Only the zero-shot version of this prompt was used, and it can be found in the appendix C.2.

[21] https://raw.githubusercontent.com/wbsg-uni-mannheim/wdc-smb/main/src/eval/table_gpt_prompt.txt

Multi-Column Prompt

As shown later, the experiments revealed that the single-column prompt did not perform satisfactorily. Therefore, the prompt from the Table GPT paper [44] was used instead. This prompt offers more instructions and enables the model to indicate whether there is a matching column in the DBPedia table for each column of the web table. This provides the model with more context and guides it towards a more accurate response. The prompt can be found in the appendix C.2. The Few Shot Prompt provides the same instructions as the Multi Column Zero Shot Prompt, but with the addition of 2 examples beforehand. This may improve the model's performance. The prompt can be found in the appendix C.2.

Sentiment Analysis

The LLM is given a sentence and specific terms to predict sentiment towards in the sentiment analysis prompts. There are two types of response: single-term and multi-term.

Single-Term Prompt

The zero-shot single-term prompt (Appendix C.3) requests the sentiment for only one term. This prompt type is the most basic. However, the LLM may struggle to understand the intended sentiment of individual terms within the sentence, instead predicting the sentiment of the sentence as a whole. The response for this prompt type is limited to 'positive', 'neutral' or 'negative', making it easy to process. The few-shot version (Appendix C.3) of this prompt has the same instructions but provides three examples beforehand.

Multi-Term Prompt

To give the LLM more context, a zero-shot multi-term prompt (Appendix C.3) will be used to request the sentiment for all terms in the sentence in one response. The LLM should respond in a JSON format that lists all terms and their respective sentiments. This will allow for easy processing of the responses. The instructions for the few-shot prompt version (Appendix C.3) remains the same, but it includes three examples before the instructions.

Price Prediction

The price prediction prompts require all independent variables to be inputted in a list format. The required response is the output price in Indian rupees. There are two types of prompts: one zero-shot prompt (refer to Appendix C.4) and one few-shot prompt (refer to Appendix C.4). The few-shot prompt includes three examples prior to the instruction. The instruction is identical to the one in the zero-shot prompt.

3.6 Baseline Analysis ML Solutions

For each task, a traditional machine learning approach was used to compare its performance to the LLM responses. Since the three tasks have different formats and value types, we had to employ different techniques. All models were run locally on a 13” 2020 MacBook Pro (2GHz Quad-Core Intel Core i5, 16GB RAM), so there were no computing costs.

Schema Matching

To complete the Schema Matching task, a Logistic Regression model was utilized. To ensure a fair comparison with LLM results, the model was provided with an equal amount of information. The model was required to predict whether each column pair corresponded or not. To achieve this, the column from table A, the column from table B, and the entire table were vectorized using a TF-IDF Vectorizer [22] from Sklearn and were then passed as separate features to the model. The entire table was considered, to ensure that the Regression Model had the same context as the LLM. Additionally, to address the unequal distribution of True and False values in the dataset, the class weight of the True values in the Logistic Regression was increased. Without the weights, the model only predicted False values, even for the training set. The tokens relevant to the two columns of an entry were extracted from the model and saved with their respective weights in a new column. This information can be used later for automated error analysis.

Sentiment Analysis

A logistic regression model was also used for the sentiment analysis task. The text and a term were encoded using TF-IDF Vectorizer and then passed to the model as separate features. The relevant tokens that appeared in the specific sentence were then extracted by the model and stored in a new column with their respective weights.

[22] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Price Prediction

The task of price prediction involves a numerical dependent variable and several numerical and categorical independent variables that must all be used for the prediction. To accomplish this, the following machine learning techniques were employed using the sklearn library in Python: Linear regression, Lasso regression, Ridge regression, ElasticNet regression, Polynomial feature regression, Random Forest Regression and Decision Tree regression. The aim was to identify a machine learning model that could achieve satisfactory performance in predicting prices while also providing a comprehensive and understandable explanation of the prediction, such as feature weights.

3.7 Evaluation Metrics

Classification

Evaluation metrics are crucial for determining the performance of classification models. These metrics assess the accuracy of predictions made by a model, particularly in binary classification problems where the outcomes are either true (positive) or false (negative). This section outlines the key metrics used in the thesis, including true positives, true negatives, false positives, false negatives, accuracy, precision, the confusion matrix, recall, and the F1 score. In the following, these evaluation metrics are first explained for the binary classification problem, where the prediction can be either "Positive" or "Negative".

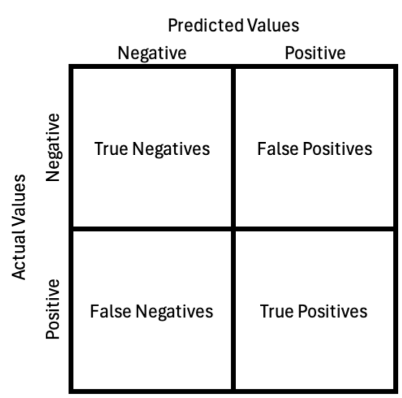

Figure 3.2: Confusion Matrix

- **True Positives (TP)**: The number of correctly identified positive cases.
- **True Negatives (TN)**: The number of correctly identified negative cases.
- **False Positives (FP)**: The number of negative cases incorrectly identified as positive. (Type I error)
- **False Negatives (FN)**: The number of positive cases incorrectly identified as negative. (Type II error)

The confusion matrix (Figure 3.2) is a table that describes the performance of a classification model. It presents the frequencies of the TP, TN, FP, and FN values, which facilitates the calculation of various performance metrics.

$$\text{Accuracy (A)} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy is a measure of the proportion of true results (both true positives and true negatives) in the total data set [86]. It provides a quick snapshot of the overall effectiveness of the model by measuring the proportion of correct predictions. However, it can be misleading in cases of imbalanced class distributions, where predicting the majority class can still lead to high accuracy and it does not differentiate between the types of errors made by the model.

$$\text{Precision (P)} = \frac{TP}{TP + FP}$$

Precision measures the proportion of true positive results in all positive predictions. It is an important metric in situations where the cost of false positives is high and helps assess the model's reliability in predicting positive outcomes. However, it does not consider false negatives. Therefore, a model can have high precision but still miss many actual positive cases. As a result, it may not be useful as the sole metric in imbalanced datasets where positive cases are rare.

$$\text{Recall (R)} = \frac{TP}{TP + FN}$$

Recall measures the proportion of true positive results among all actual positives. It is crucial in scenarios where missing a positive case (false negative) can have serious consequences, such as in medical diagnosis. However, a model can achieve high recall by predicting more cases as positive, which increases the risk of false positives.

$$\text{F1 Score (F1)} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2xTP}{2xTP + FP + FN}$$

The F1 score provides a balance between precision and recall by being the harmonic mean of the two [8]. It is the main metric used to evaluate results in this thesis, as a high F1 score indicates effective solutions that mitigate both type I and type II errors. The F1 score is particularly useful when the class distribution is imbalanced. However, it may not fully capture the trade-offs in specific applications where either precision or recall is more important.

In trinary (or multi-class) classification, such as the sentiment analysis in this thesis, outcomes are categorized into three or more classes. To apply the aforementioned metrics to trinary classification, the most common approach is to evaluate each class against the rest, known as the one-vs-all method. For each class, metrics are calculated by considering one class as positive and the rest as negative. This process is then repeated for each class. The final metric can be the macro average or weighted average of the individual metrics for each class. Macro average calculates the average F1 score by treating each class equally, while weighted average considers the contribution of each class based on the number of examples. The choice between macro average and weighted average depends on the importance of each class and whether the dataset is imbalanced or not. In this thesis, the weighted averages will be evaluated for the sentiment analysis task, accounting for an unbalanced number of sentiments. The confusion matrix can also be extended to support trinary or multi-class classification. This results in a larger table that includes true and false positives and negatives for each class against every other class.

Regression

Various metrics are used to evaluate the performance of regression models. These metrics provide different insights into the accuracy and reliability of the predictions. The most significant metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R^2) [65]. In the formulas provided, the variables represent the following:

- y_i: The actual observed values from the dataset.
- $\hat{y}_i$: The values predicted by the regression model.
- $\overline{y}$: The mean (average) of the actual observed values.
- n: The number of observations or data points.

MAE is the average of the absolute errors between the predicted values and the actual values. It gives an idea of how big the errors are on average. The MAE is calculated as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MSE is the average of the squared differences between the predicted values and the actual values. It places a larger penalty on larger errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

RMSE is the square root of MSE and similarly provides a measure of the magnitude of error.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

R^2 is the proportion of the variance in the dependent variable that can be predicted from the independent variables. It ranges from 0 to 1, with 1 indicating a perfect fit.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

Each of these metrics can be evaluated based on the specific context of the problem at hand. The MAE is particularly useful when you want to avoid punishing larger errors (as it doesn't square the error term). The MSE and RMSE are more sensitive to outliers than the MAE. An advantage of the RMSE over the MSE is that its units are the same as the units of the dependent variable. The R^2 metric indicates the proportion of the variance in the dependent variable that is predictable from the independent variables, with a value closer to 1 indicating a better fit. Although a higher R^2 value signals a better fit for the model, it is important to note that a low R^2 value does not necessarily mean that the model is inadequate.

3.8 Error Analysis

In order to gain a deeper insight into the decision-making of LLMs and to test whether LLMs can be used to identify and cluster errors from their own explanations and the explanations of machine learning models, the LLMs were prompted in the following areas:

1. Confidence level of deicsion / prediction
2. Structured and Unstructured explanations
3. Error class clustering

Confidence level

The LLM was provided with the prior conversation of a decision, so the prompt input and its prediction output, and was tasked to provide its confidence in the decision it had previously made. Therefore, the prompt input was labelled as human input and the prediction output was labelled as AI output, allowing the model to distinguish between the two. The task of expressing confidence in a particular format was then appended as human input. The resulting prompt template had the following format, inspired by the prompt used by Burkhardt [18]:

```
confidence = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """{ai_answer}"""),
    ("human", """Now provide a confidence score for your decision.
         100% referring
    to full confidence. Take into account all relevant apsects of
        your decision. Please provide your answer exactly in the
        following format and do not return anything else.
Confidence: <value>%""")
])
```

For the 3 tasks, the prompts were created based on this format, but each was slightly adapted to take into account the format of the previous input. For the price prediction task, the model was told that a correct prediction was a prediction that was within $\pm 20\%$ of the actual value. For the schema matching task, the model was told the specific column pair for which it should provide confidence and also whether it predicted that they matched or not. The template for the Price Prediction prompt can be found in Appendix C.4, the template for the Schema Matching prompt can be found in Appendix C.2 and the Sentiment Analysis prompt can be found in Appendix C.3.

A *Receiver Operating Characteristic (ROC)* was used to evaluate the confidence scores. A ROC curve is a graphical representation that shows the diagnostic ability of a binary classifier system as its discrimination threshold is varied [1]. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. *Area Under the Curve (AUC)* refers to the area under the ROC curve. This metric provides a single scalar value to summarize the performance of a binary classifier, independent of any particular threshold. The AUC ranges from 0 to 1, with an AUC of 1 indicating a perfect model that has a perfect true positive rate and a false positive rate of 0 for all possible threshold values. On the other hand, an AUC of 0.5 indicates a model with no discriminative ability whatsoever, equivalent to random guessing.

Structured and Unstructured Analysis

In order for the LLM to be able to recognise errors and group them into different classes, it must be provided with additional explanations of how the decision was made. Therefore, the LLM is again presented with the previous conversation including the input prompt and its own prediction, and it is tasked with providing an explanation for this prediction in two formats:

Structured Explanation

In the structured explanation, the model is asked to provide an explanation in a structured, quantifiable format. In this case, the model was asked to provide the explanation in JSON format. An example output for the price prediction, where the model was asked to identify the most important attributes of the car and provide an importance score, is shown below:

```
[
{"attribute":"year","importance":"0.30","value":"2008"},
{"attribute":"km_driven","importance":"0.25","value":"110000"},
{"attribute":"fuel","importance":"0.15","value":"Petrol"},
{"attribute":"owner","importance":"0.10","value":"Fourth & Above
    Owner"},
{"attribute":"max_power","importance":"0.20","value":"68.05 bhp"}
]
```

The full prompt template for the Price Prediction task can be found in Appendix C.4. No logical structured explanation format was found for Schema Matching. For the sentiment analysis, the model was asked to output the adjectives in the sentence that it considered most important, and also to append meanings. The full prompt template can be found in the appendix C.3.

Unstructured Explanation

In the unstructured explanation runs, the model was asked to openly explain its decision in no particular format. The prompt was always structured as follows

```
unstructured_analysis = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
```

```
    ("ai", """{ai_answer}"""),
    ("human", """Now explain concisely how you made your
       prediction and explicitly mention the words or word groups
        that had a high influence on your decision.""")
])
```

Again, there are slight differences due to the specific output formats of the different tasks. The Schema Matching template can be found in Appendix C.2, the Sentiment Analysis prompt in Appendix C.3 and the Price Prediction prompt in Appendix C.4.

Automated Error Class Clustering

This part evaluates the ability of the LLM to automatically detect and cluster errors. This is done for the runs and explanations generated by the LLM itself (LLM runs), but also for the runs performed with the classical machine learning models (ML runs).

An error class prompt will always have the following structure: First, the LLM is given a prompt, asking it to identify and cluster the errors it has been given. Then it is given 20 cases that it predicted incorrectly and 5 that it predicted correctly for reference. Each case contains the original prompt, the prediction, the explanation (structured, unstructured or ML weights) and the actual correct answer to the task. For example, for Schema Matching it looks like this:

```
{Instructions}

Wrong Decisions:
Task 1:
Prompt: {prompt}
Answer: {answer}
Explanation: {explanation}
Correct Answer: {correct answer}

[...]

Correct Decisions (Only as a reference):
Task 21:
Prompt: {prompt}
Answer: {answer}
Explanation: {explanation}
Correct Answer: {correct answer}

[...]
```

LLM runs

The error class clustering will be performed on the LLM runs and as explanations the unstructured and structured explanations will be fed in. It is also explained to the LLM that the prediction and the provided explanation come from an LLM itself. The schema matching instruction can be found in Appendix C.2, the sentiment analysis instruction can be found in Appendix C.3 and the price prediction instruction can be found in Appendix C.4.

ML runs

The error class clustering will also be performed on the ML runs and as explanations the weights for the tokens that are relevant for this specific entry from the machine learning model will be fed in. It is also explained to the LLM that the prediction and the explanation provided come from a classical machine learning model. The schema matching instruction can be found in Appendix C.2, the sentiment analysis instruction in Appendix C.3 and the price prediction instruction in Appendix C.4.

Chapter 4

Experimental Evaluation

4.1 Schema Matching

Dataset	#Table Pairs	#Pos	#Neg	Pos/Neg Ratio
Test Set (Downsampled)	28	79	371	0.21
Test Set	89	209	1056	0.20
Training Set	71	177	878	0.20
Validation Set	18	59	263	0.22

Table 4.1: T2D-SM NH dataset statistics

4.1.1 Baseline Analysis

The table 4.1 displays the number of entries in each dataset. The training, validation, and test datasets were obtained from the schema matching benchmark website[1]. The test set has been downsampled, and the Pos/Neg Ratio column indicates a significantly higher number of negative entries than positive ones, which must be considered in future evaluations.

The summary table 4.2 presents the LLM results for the Zero Shot, Fix Few Shot, and Dynamic (Dyn.) Few Shot prompts on the GPT-3.5-turbo-0125 (GPT-3.5-t) and the GPT-4-0125-preview (GPT-4-p) models. Moreover, it also shows the ML results for the Logistic Regression (Log. Reg.) model. The table evaluates Accuracy (Acc.), Precision (P), Recall (R), F1 value and the confusion matrix.

[1] https://webdatacommons.org/structureddata/smb/#toc5

Prompt	**Model**	**Acc.**	**P**	**R**	**F1**	**Confusion Matrix**
Single Column:						
Zero-Shot	GPT-3.5-t	0.59	0.14	0.27	0.19	[246, 125] [58, 21]
Zero-Shot	GPT-4-p	0.84	0.71	0.13	0.22	[367, 4] [69, 10]
Multi Column:						
Zero-Shot	GPT-3.5-t	0.84	0.64	0.23	0.30	[361, 10] [61, 18]
Zero-Shot	GPT-4-p	0.88	0.88	0.38	0.52	[367, 4] [49, 30]
Fix-Few-Shot	GPT-3.5-t	0.85	0.61	0.35	0.45	[353, 18] [51, 28]
Fix-Few-Shot	GPT-4-p	0.90	0.87	0.49	0.63	[366, 5] [40, 39]
Dyn.-Few-Shot	GPT-3.5-t	0.85	0.67	0.28	0.39	[360, 11] [57, 22]
Dyn.-Few-Shot	GPT-4-p	0.91	0.91	0.53	0.67	[367, 4] [37, 42]
ML Solution:						
-	Log. Reg.	0.46	0.18	0.61	0.28	[157, 214] [31, 48]

Table 4.2: Baseline analysis summary Schema Matching

Single Column

An examination of the single column results of Table 4.2 reveals that the models demonstrated significantly different accuracies when used in a zero-shot configuration. GPT-3.5-t achieved an accuracy (Acc.) of 0.59, which is significantly lower than GPT-4-p's 0.84. However, both achieved comparable F1 scores of around 20%. The confusion matrix shows that the GPT-3.5-t approach has a significant number of False Positives, resulting in a lower Precision. On the other hand, the GPT-4-p model has a higher number of False Negatives, leading to a lower Recall value. However, the analysis of the single column approach in the table is limited to the presented figures due to its unsatisfactory performance. The F1 scores of 0.19 and 0.22 for the GPT-3.5-t and GPT-4-p models, respectively, illustrate the inferiority of the single-column approach. Regarding the GPT-3.5-t approach, it is probable that the LLM did not comprehend the context and provided a random response when asked about the matching of two columns. Regarding the GPT-4-p approach, the LLM indicated that for most column pairs, two columns do not match. This suggests that the prompt does not provide the appropriate context for the LLM. This led to the decision to investigate a multi-column approach, where the LLM is not asked specifically if two columns match but can find all column matches for the whole table freely.

Multi Column

In the multi-column part of the analysis, there is a notable improvement in performance metrics across all models and methods compared to the single-column approach. The Zero-Shot configuration with both GPT-3.5-t and GPT-4-p shows significant improvements in Accuracy, Precision, Recall and F1 scores when moving from the single-column to the multi-column approach. In particular, the GPT-4-p model shows a remarkable improvement in performance with an Accuracy of 0.88, Precision of 0.88, Recall of 0.38 and F1 score of 0.52 compared to an F1 score of just 0.22 for the Zero-Shot GPT-4-p run. These metrics indicate that the GPT-4-p model is significantly more adept at understanding and matching multiple columns simultaneously than in a single-column pair matching scenario, where its performance was already ahead of GPT-3.5-t.

The Fix-Few-Shot and Dynamic-Few-Shot configurations further improve these metrics, with the Dynamic-Few-Shot configuration of GPT-4-p achieving the highest scores across all models and configurations: Accuracy of 0.91, Precision of 0.91, Recall of 0.53 and an F1 score of 0.67. These improvements highlight the effectiveness of few-shot learning in improving the model's ability to accurately identify and match column pairs. This can most likely be explained by the following: Looking at the confusion matrix of the two Zero-Shot runs, it can be inferred that the model tends to have a bias towards matching too many columns to None instead of another column, no matter if it does that correctly or not. For the Zero-Shot GPT-3.5-t run, only 28 out of 450 total predictions were a matching prediction to another column by the LLM, making up only about 6.2% of the total predictions. For the Zero-Shot GPT-4-p this percentage rises to 7.6% of the total predictions being matching predictions. However, the percentage increases again for the Few-Shot runs because the examples show the model that it can and should make more matching predictions, rather than just matching everything to None. Without the examples, it is assumed that often the model did not understand that even if two columns do not have exactly the same words or numbers, or are not spelled the same, for example with NYC and New York, they can still match. Through the examples it did understand this, which resulted in the percentage of 'positive' match predictions of the total going up to 10.2% for the Dynamic Few-Shot GPT-4-p run, showing that the bias has been decreased, which is also evidenced by significantly increased recall values. Interestingly, there is a decrease in performance from the Fix-Few-Shot GPT-3.5-t to the Dynamic-Few-Shot GPT-3.5-t run in F1 values, but a corresponding increase from Fix to Dynamic for GPT-4-p, which cannot be fully explained. It appears that the randomly selected Fix examples help GPT-3.5-t more than the more dynamic examples of the Dynamic prompts, to which GPT-3.5-t does not seem to adapt and therefore the bias increased again, also evidenced by a decreasing recall value from 0.35 to 0.28. For other Fix examples, however, the results may be different.

ML Solution

The logistic regression achieved an accuracy of 0.46, a precision of 0.18, a recall of 0.61 and an F1 score of 0.28. These results indicate a markedly different performance profile, particularly in terms of precision and recall, compared to the LLM runs. The relatively high recall indicates that the logistic regression model tends to identify a larger proportion of true positive matches as such. It has the highest recall of all runs for the schema matching tasks. The confusion matrix for the Logistic Regression model further illustrates this dynamic, with a significant number of false positives (214) compared to true positives (48). This imbalance contrasts sharply with the confusion matrices observed for the LLM approaches, especially in multi-column scenarios, where improvements in both precision and recall were notable. This effect can be attributed to the fine-tuning of the model on class_weight. The class_weight parameter was set to a ratio of 1.5 to 1 for positive to negative values, overweighting the true values compared to the original dataset ratio and the C value was set to 1. This was done because without setting this parameter, the model predicted 100% of matches as false, even on the dataset it was trained on. This explains the strong bias of the model towards predicting matching columns compared to the LLM runs. However, this comes at the expense of Precision, as evidenced by the low value, suggesting that the model also incorrectly labels a significant number of non-matching columns as matches. This imbalance between precision and recall culminates in a moderate F1 score, which is acceptable as logistic regression is not designed to solve a task such as schema matching perfectly.

Costs

Due to the larger number of input tokens resulting from the table representations, the costs were more significant than for other tasks. This was particularly the case for the single-column run, where each pair of columns to be checked required an extra prompt containing the entire tables. Ultimately, this was also an important factor in only testing the zero-shot prompt and not the few-shot prompts, as these would have been expensive, especially on GPT-4-p, while presumably resulting in worse performance than its multi-column counterpart. While the GPT-3.5-t single column runs cost $0.15, the GPT-4-p run already cost $2.50. In comparison, the multi-column runs cost only $0.008 and $0.15 for GPT-3.5-t and GPT-4-p respectively, as only one prompt per table in dataset was required.

Findings & Discussion

In conclusion, it was decided that the logistic regression model performed satisfactorily for further error analysis, as it also provides weights for the tokens of an entry, which can be used later for explanation. Furthermore, the multi-column LLM runs showed good performance while still providing enough errors for further analysis. Interestingly, the Zero-Shot GPT-4-p run only achieved an F1 value of 0.52, while the WDC Schema Matching Benchmark [21] achieved an F1 value of 0.75 for the Table GPT prompt. This difference can be attributed to the use of different models, as the Schema Matching Benchmark seems to use a model from the normal GPT-4 series, and the prompt used here was also slightly modified.

4.1.2 Error Analysis

For the following steps in the schema matching error analysis, 4 runs have been selected. The first three are the multi-table runs carried out with GPT-4-p using the Zero-Shot, Fix-Few-Shot and Dynamic-Few-Shot prompts. The fourth run was the Logistic Regression run, which will be compared with the other three.

Confidence Analysis

The confidence analysis was conducted using GPT-3.5-t due to cost considerations and was performed on the 3 LLM runs. To aid comprehension, a ROC curve with an AUC value was generated for each run, which can be seen in Figure 4.1. All confidence values for the non-match predictions were inverted to ensure a logical analysis, and the confidence was used as a threshold. The greater the area under each curve, the more accurate the test. The AUC value indicates the predictive ability of the test. A value close to 0.5 suggests poor performance, equivalent to random guessing. The dynamic few-shot run and fixed few-shot run tests performed well in assessing their confidence, with AUC values of 0.69 and 0.70 respectively. However, the LLM did not perform well in assessing confidence for the Zero-Shot run. Upon examining the curve and the AUC value of 0.55, it is evident

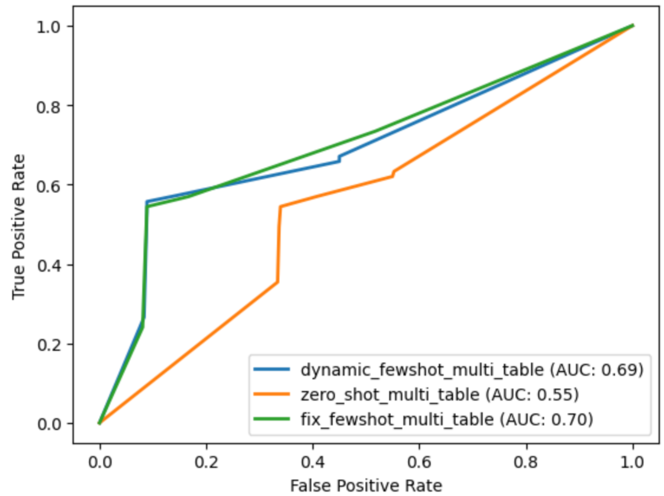

Figure 4.1: ROC Curve Schema Matching

that the LLM performed only marginally better than random guessing. This difference in performance appears to be attributed to the LLM's better performance when provided with examples from the few-shot prompts that were not available in the zero-shot run. It is important to note that this analysis does not account for the class distribution, which could result in over- or underestimation of the classifier.

Unstructured Explanations

A logical method for generating structured explanations for the Schema Matching task could not be found. Therefore, only unstructured explanations were generated using GPT-4- p, which is believed to produce higher quality explanations. However, only 25 false and 5 correct predictions were used to generate explanations due to the high costs, particularly for the few-shot prompts where multiple large tables increase the token size. This set size will also be used for the error class generation. The false and correct predictions were randomly sampled.

To assess the quality of the unstructured explanations generated, it can be stated that each explanation highlights the decision-making process based on the content and context of the data in the compared columns. The LLM produced similar explanations in all three runs, as they were generated by the same model. The LLM stated that matching columns typically originate from the same domain or have the same data type, and usually began with an introductory sentence. The argument was supported by examples presented in a structured list format, allowing for easy comprehension. For non-matching columns, the LLM followed a similar approach, providing examples and supporting the argument with data-type or domain information. Additionally, the LLM occasionally identified a match between one of the columns and another column in the table, demonstrating a strong contextual understanding of the task. The explanation concluded with a summary paragraph or sentence, where the main reason for matching or not matching the columns was again provided. However, there was no noticeable difference in the quality of explanations for the various types of prompts. All unstructured explanations for this task can be found in the appendix A in the Datasets/Evaluations/Schema Matching/Error_Analysis/structured_unstructured folder.

Error Classes

The LLM was then asked to create and cluster error classes for each run. The GPT-4-p model was used for this, as Burkhardt [18] has already shown in another task that the quality of the responses was significantly better for GPT-4-p. As indicated in the prompt in the Methodology section, no limit on the number error classes was given. The results are shown in the table A.1 (in Appendix A.1). The maximum amount of error classes that was generated was six. Although the table provides an overview of the error classes, the LLM responses provided more explanation and context. For most runs, the LLM listed each error class and then explained it in more detail. It also listed the number of occurrences and the tasks in which it thought that particular error was made. The error classes for the logistic regression can be found in the appendix A.1. Here, the LLM provided explanations, occurrence counts, and specific examples for each error class. However, the error classes do not appear to be well-analyzed in terms of fitting a logistic regression model, despite the LLM being informed of the logistic regression model and its associated weights for tokens. The text appears to provide general potential error classes, such as mismatched content types, but does not explain how logistic regression may have produced an error. Additionally, there is no mention or reference to the weights provided as an explanation. Therefore, it could be concluded that the model's explanations in terms of weights did not add significant value or lead to better or more specific error classes. The error classes generated by the logistic regression run are similar to those generated by the normal LLM runs. This suggests that the LLM provides more general potential error classes rather than specific cases and explanations. However, the error classes provided for the logistic regression run are disjoint and of generally high quality. The full error class answers for the Zero-Shot run can be found in appendix A.1, the error class answer for the Fix-Few-Shot run can be found in appendix A.1, and the error class answer for the Dynamic-Few-Shot run can be found in appendix A.1. Unfortunately, the

LLM did not provide specific occurrences or occurrence counts for the Fix-Few-Shot run. Upon examining the error classes and their counts for the other runs, it can be concluded that there are significantly fewer occurrences when transitioning from Zero-Shot to Dynamic-Few-Shot and some error classes disappear. The *'Mismatch in Data Type or Format'* error was found five times in both runs, but aside from that, the occurrence count decreased significantly. The *'Incorrect Matching'* error class, which occurred in both Zero-Shot and Dynamic-Few-Shot, decreased from 7 to 1 occurrence. The other two error classes in Zero-Shot, with 3 and 5 occurrences respectively, disappeared completely. In addition, 3 more specific and new error classes appeared in the Dynamic-Few-Shot run, each with 1 occurrence, such as *'Mismatch in Specificity and Context of Geographical Information'*. The *'Mismatch in Data Type or Context'* error class was found in all four runs, with a slight deviation for the logistic regression. It should be noted that for the Logistic Regression and the Zero-Shot run, the LLM found significantly more occurrences than for the Dynamic-Few-Shot run. In conclusion, it seems that the LLM is not able to find error classes that are specific to the logistic regression and sometimes fails to provide occurrence counts. In all runs, the LLM did not categorise any correct examples as errors. In the Dynamic-Few-Shot run, however, not all false examples were categorised as errors. However, the LLM appears to accurately identify error classes in LLM runs. It recognized that the Few-Shot runs result in the LLM identifying fewer occurrences of error classes, as some errors of the LLM have probably been prevented due to the provided examples and context. However, the LLM appears to accurately identify error classes in LLM runs, while Few-Shot runs result in the LLM identifying fewer occurrences of error classes.

4.2 Sentiment Analysis

Dataset	**#Pos (ratio)**	**#Neg (ratio)**	**#Neu (ratio)**	**#Total**
Test Set (Downsampled)	137 (0.30)	108 (0.24)	206 (0.46)	451
Test Set	400 (0.30)	329 (0.25)	607 (0.45)	1336
Training Set	3380 (0.30)	2764 (0.25)	5042 (0.45)	11186
Validation Set	403 (0.30)	329 (0.25)	607 (0.46)	1332

Table 4.3: MAMS ATSA dataset statistics

Prompt	**Model**	**Acc.**	**P Wei**	**R Wei**	**F1 Wei.**	F1 Neg.	F1 Neu.	F1 Pos.
Single Term:								
Zero-Shot	GPT-3.5-t	0.59	0.60	0.59	0.57	0.66	0.46	0.67
Zero-Shot	GPT-4-p	0.59	0.61	0.59	0.57	0.68	0.44	0.65
Fix-Few-Shot	GPT-3.5-t	0.48	0.51	0.48	0.40	0.58	0.14	0.61
Fix-Few-Shot	GPT-4-p	0.60	0.66	0.60	0.57	0.66	0.44	0.66
Dyn.-Few-Shot	GPT-3.5-t	0.63	0.63	0.63	0.63	0.65	0.59	0.67
Dyn.-Few-Shot	GPT-4-p	0.58	0.62	0.58	0.56	0.65	0.44	0.65
Multi Term:								
Zero-Shot	GPT-3.5-t	0.64	0.66	0.64	0.62	0.72	0.52	0.68
Zero-Shot	GPT-4-p	0.67	0.69	0.67	0.67	0.68	0.64	0.71
Fix-Few-Shot	GPT-3.5-t	0.63	0.65	0.63	0.61	0.70	0.53	0.66
Fix-Few-Shot	GPT-4-p	0.65	0.67	0.65	0.64	0.68	0.59	0.68
Dyn.-Few-Shot	GPT-3.5-t	0.63	0.65	0.63	0.62	0.68	0.55	0.68
Dyn.-Few-Shot	GPT-4-p	0.67	0.67	0.67	0.66	0.66	0.63	0.71
ML Solution:								
-	Log. Reg.	0.65	0.65	0.65	0.64	0.64	0.73	0.53

Table 4.4: Baseline summary of weighted averages (Wei.) Sentiment Analysis

4.2.1 Baseline Analysis

The table 4.3 gives an overview of the different datasets and their negative, neutral and positive entries. It is noteworthy that the ratio of neutral to total entries is significantly higher (0.45) than for positive and negative entries, which could be important for the evaluation. Table 4.4 gives an overview of the different runs of the sentiment analysis. It shows the achieved weighted averages of Accuracy (Acc.), Precision (P), Recall (R) and F1 value over the 3 categories on a one vs all basis. It also shows the individual F1 values for Negative (Neg.), Neutral (Neu.) and Positive (Pos.) predictions.

Single Term

In the Zero-Shot configuration, both the GPT-3.5-t and GPT-4-p models show almost identical performance metrics, with weighted averages for Accuracy, Precision, Recall and F1 all hovering around the 0.58 mark. This parity suggests that without additional contextual training or examples, both models have similar baseline abilities to understand and classify individual terms within these sentiment analysis tasks. The F1 scores for negative, neutral and positive sentiments further indicate that both models are slightly more adept at identifying negative (F1 Neg.) and positive (F1 Pos.) sentiments than neutral (F1 Neu.) sentiments, a trend that continues for most other runs.

Moving to the Fixed-Few-Shot and Dynamic-Few-Shot configurations, the models show various improvements in their ability to classify sentiments more accurately. In particular, the Dyn.Few-Shot configuration for GPT-3.5-t marks an improvement, achieving an F1 score of 0.63. This configuration also shows a marked improvement in the identification of neutral sentiments (F1 Neu. of 0.59), which is the main reason for the improved weighted F1, suggesting that this configuration allows the model to better capture the nuances of neutral sentiment classification. However, there is no obvious trend by changing the prompts or models, as for example the Dyn.Few-Shot GPT-4-p configuration performs worse than even the zero-shot prompt runs.

Multi Term

In the multi-term configurations, there is a noticeable improvement in the performance metrics of all models compared to the single-term analyses. In particular, the zero-shot configuration for GPT-3.5-t shows an increase in accuracy, precision, recall and weighted F1 score, with a significant improvement in negative and neutral F1 scores compared to its single-term counterpart. Overall, there appears to be no noticeable effect of the few-shot prompt compared to the zero-shot runs in the multi-term prompt scenarios. The weighted F1 scores for the fix few-shot scenario are both worse than their zero-shot counterparts. The only trend that can be observed is that GPT-4-p runs achieve better weighted F1 scores than their GPT-3.5-t counterparts for each prompt type, but the difference is not very significant. The difference is mainly due to improved neutral F1 scores. Overall, these results suggest that a multi-term prompt leads the model to better understand that there may be several different sentiments towards different terms in a sentence. By telling it to evaluate all terms in a prompt, it can better understand the different cues in a sentence and that they are only directed to certain terms, whereas in a single term setting the LLM can take all these cues into account for one term. GPT-4-p seems to understand this context even better than GPT-3.5-t. However, as is the case for the single term scenarios, the relatively high performance of the models in identifying negative and positive sentiments, as opposed to neutral sentiments, suggests that there is a distinct feature in sentiment-laden language that the LLM is able to capture more effectively than expressions of neutrality. This could be due to the more explicit nature of the language used to convey positive and negative sentiments, as opposed to the subtlety often present in neutral expressions.

ML Solution

The logistic regression model shows a competitive performance with a weighted accuracy, precision and recall of 0.65 and a weighted F1 average of 0.64. In particular, it outperforms the LLM configurations in identifying neutral sentiment, as evidenced by the highest F1 score for neutral sentiment (F1 Neu. of 0.73), while underperforming in positive classification (F1 of only 0.53) compared to the LLM runs. This highlights the effectiveness of traditional machine learning models in certain aspects of sentiment analysis, particularly in the nuanced detection of neutrality in individual terms. The low effectiveness in predicting positive values could be due to the fact that logistic regression is not able to understand and take into account the grammatical meaning of sentences, which often already signals a positive sentiment to humans and LLMs, but can probably only recognise and weigh positively connotated adjectives and verbs to assess sentiment. Furthermore, there may simply be more obvious negatively connotated words than positively connotated words, where the meaning can also be derived from the grammar, explaining the difference in F1 scores. This could be the subject of further analysis. The parameters of the logistic regression model were set to C=1, penalty = 'l2' and solver='saga', derived from fine-tuning with the training set. Fine-tuning with the validation set could lead to further improvements by avoiding over-fitting, but was not performed here as the validation set was not pre-processed.

Costs

The cost of the sentiment analysis prompts did not play a significant role in deciding which scenario to choose for the error analysis. A single term GPT-3.5-t run cost around $0.05 and a single term GPT-4-p run cost $0.75. A single term GPT-3.5-t run cost around $0.10 and a single term GPT-4-p run cost $1.50. As the logistic regression was run locally, no costs were incurred.

Findings & Discussion

Due to the smaller differences in F1 scores between the single and multi term questions compared to other tasks, it was ultimately decided to perform the error analysis on the single term runs. This was mainly due to the fact that the answer structure of the questions makes it easier to mark a prompt and the corresponding answer as correct or incorrect and to get explanations from the LLM, whereas this would be more nuanced on the multi term runs. As a result, error analysis is much easier to perform on the single term runs. Logistic regression was also used for error analysis as its performance is satisfactory, and it can provide explanations in terms of weights.

4.2.2 Error Analysis

Four runs have been selected for the following steps in the sentiment analysis error analysis. The first three runs were single-term runs conducted with GPT-4-p using the Zero-Shot, Fix-Few-Shot, and Dynamic-Few-Shot prompts. The fourth run was a Logistic Regression run will be compared with the other three.

Confidence Analysis

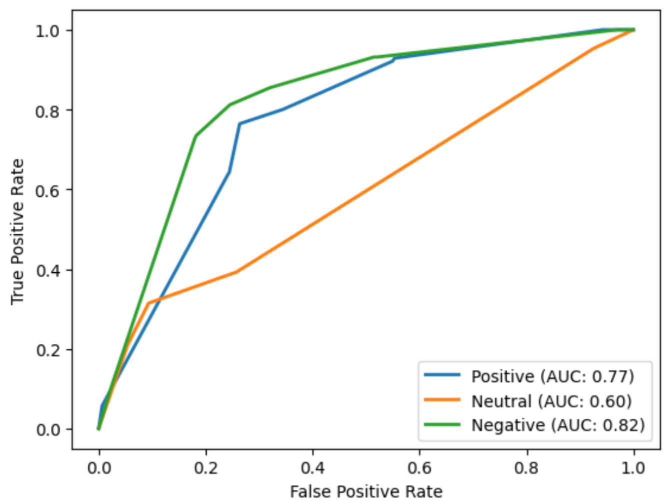

Figure 4.2: ROC Curve Dynamic-Few-Shot run Sentiment Analysis

The confidence analysis for the schema matching was conducted using GPT-3.5-t due to cost considerations. It was performed on three LLM runs. To account for the multi-class nature of the sentiment analysis prediction, a separate figure was created for each run. Figure 4.2 displays the ROC curves for the positive, neutral, and negative classifications on a one vs. all basis. For example, for the Positive ROC curve, the positive values were transformed to a 1, while the other two values were counted as 0. For this to work, all confidences for 0 predictions have been inverted. The figure above shows that the tests for Positive and Negative values are relatively accurate with AUC values of 0.77 and 0.82 respectively, while the Neutral AUC value of 0.60 suggests less accuracy and more random prediction patterns. This pattern repeats for the ROC curves of both other runs, which can be found in Appendix A.2. In general, it can be concluded that LLMs perform well in identifying positive and negative sentiment towards terms in sentences based on confidence scores. However, they appear to have difficulty with neutral sentiments.

This is understandable, as in aspect-based sentiment analysis, the line between neutral and positive or neutral and negative is often not entirely clear, and even a human evaluator could have difficulty with it.

Unstructured Explanations

Unstructured explanations were requested for all three LLM runs, and GPT-4-p was used as the model. However, due to the high costs, especially for few-shot prompts, only a sample of 50 correct and 50 incorrect predictions was taken to generate explanations. This should not pose a problem as the error analysis only requires 20 incorrect and 5 correct predictions. The CSV file containing unstructured explanations can be found in the code appendix B under the folder Datasets/Evaluations/Sentiment Analysis/Error_Analysis/structured_unstructured. The quality and structure of the answers did not differ significantly across the different runs. An answer typically consists of a few clear and concise sentences that logically explain the decision for a specific sentiment and list important words in the sentence that led to the decision. Explanations may also include the sentiment of the whole sentence or the absence of sentiments towards a specific term. For example, when evaluating the sentiment towards the term "onions" in the sentence *"The smoked salmon appetizer was pretty good (it comes with goat cheese, capers and onions)"*, one possible answer could be: *"The prediction was based on the absence of any specific sentiment towards 'onions' in the given text. The overall sentiment of the sentence is positive, as indicated by the phrase 'pretty good,' but this sentiment is directed towards the smoked salmon appetizer as a whole, rather than any of its individual components, such as goat cheese, capers, or onions. As there are no words or phrases that express a sentiment towards 'onions' specifically, the decision was made to classify the sentiment as 'neutral'"*

Structured Explanations

To generate structured explanations, it was found that allowing the LLM to list the adjectives that were important for the sentiment towards the specific term and providing an importance score was the most effective method. The full answers can also be found in appendix B under the folder Datasets/Evaluations/Sentiment Analysis/Error_Analysis/structured_unstructured. This approach generally worked well, as the LLM identified relevant adjectives and their impact on the term. However, in cases where no adjectives were found, the LLM sometimes returned whole phrases instead of an empty list. The LLM consistently produced valid JSON format without any errors. The mean importance score for all attributes is 0.55, with a standard deviation of 0.2784 and a variance of 0.0775. Further summary of the attributes would not be meaningful due to their high diversity.

Error Classes

	Zero-Shot	**Fix-Few-Shot**	**Dyn.-Few-Shot**
Unstructured Explanations			
Class 1	Misinterpretation of Contextual Sentiment (14)	Misinterpretation of Context or Detail (20)	Misinterpretation of Contextual Sentiment (19)
Class 2	Neutral Misclassification (4)	Incorrect Sentiment Polarity (-)	Incorrect Sentiment Polarity (1)
Class 3	Overgeneralization of Sentiment (2)	Overgeneralization (3)	-
Structured Explanations			
Class 1	Misinterpretation of Contextual Sentiment (4)	Misinterpretation of Context or Negation (7)	Misinterpretation of Context or Modifier (20)
Class 2	Incorrect Sentiment Attribution (14)	Incorrect Sentiment Attribution (14)	Incorrect Sentiment Polarity (1)
Class 3	Failure to Recognize Neutral Sentiment (3)	Overlooking Specific Details or Misjudging Importance (1)	-

Table 4.5: Error Class Overview for LLM runs with # of occurences for Sentiment Analysis

	Logistic Regression
Class 1	Misinterpretation of Context or Negation (5)
Class 2	Incorrect Sentiment Polarity (4)
Class 3	Failure to Recognize Positive Aspects (7)
Class 4	Failure to Recognize Neutral Aspects (1)
Class 5	Incorrect Neutral Sentiment Assignment (3)

Table 4.6: Error Class Overview for Log. Reg. with # of occurrences for Sentiment Analysis

Error classes were created for each of the three LLM runs using both structured and unstructured explanations with the GPT-4-p model. The model was provided with 20 incorrect and 5 correct predictions. Table 4.5 shows the error classes and occurrence counts for both structured and unstructured explanations. Table 4.6 displays the error classes generated by the Logistic Regression and their occurrences. The logistic regression weights for all words in the sentence were provided as an explanation. The tables provide an overview of the error classes, but the LLM often provided further explanation of the error classes, and sometimes why the LLM assigned a particular error to a particular error class. The full error class responses can be found in Appendix A.2.

Overall, it can be concluded that the LLM sorted only the incorrect examples into the error classes and provided occurrence counts in most cases. Although the count of error classes for all six LLM runs ranges from two to three, the Logistic Regression reported five error classes. This indicates that the LLM was able to identify more potential sources of error when provided with information that the prediction was done by a logistic regression and the assigned weights as explanations. When comparing the error classes for structured and unstructured explanations, it is evident that the error class 'Misinterpretation of Contextual Sentiment' occurs in every run. However, for the Zero-Shot and Fix-Few-Shot runs, the count of this error class was significantly higher when the LLM was provided with unstructured explanations than with structured explanations. It can also be concluded that this error class is very generic and applies to most, if not all, errors. Also, error classes that occur frequently, such as 'Incorrect Sentiment Polarity' or 'Incorrect Sentiment Attribution', are too generic and do not provide any additional information on the errors. This indicates that the LLM is not fully capable of providing specific error classes in the case of sentiment analysis with LLM explanations, but rather provides several generic and non-disjoint error classes. Some error classes, such as 'Overgeneralization of sentiment' or 'Overlooking Specific Details or Misjudging Importance', occur less frequently but provide a more specific picture of the errors. It appears that these error classes are derived from the explanations, which shows that the LLM can identify them. When using a general prompt for error classification like in this case, the LLM seems unable to provide meaningful and valuable error classes for sentiment analysis. Therefore, it could be interesting to use more specific error class prompts to guide the LLM in providing better error classes in future research. The error classes for Logistic Regression appear to have better quality. An error class such as 'Misinterpretation of Context or Negation' accurately describes the issues that a logistic regression model may encounter when performing sentiment analysis. However, there are still more general and non-disjunct classes such as 'Incorrect Sentiment Polarity' present. Overall, it can be concluded that the occurrence count for logistic regression is more balanced than that of LLM runs, where most errors are classified into one generic error class.

4.3 Price Prediction

Dataset	#Entries (ratio of original)	Price Avg.	Price Std.
Original Dataset	8128 (1.00)	638,271	806,253
Test Set (Downsampled)	400 (0.05)	624,817	846,695
Test Set	1626 (0.20)	636,732	809,869
Training Set	5201 (0.64)	639,316	808,574
Validation Set	1301 (0.16)	636,023	792,931

Table 4.7: Vehicle dataset statistics

4.3.1 Baseline Analysis

Prompt	Model	MAE	MSE	RMSE	R^2
LLM Results:					
Zero-Shot	GPT-3.5-t	101,752	49,843,281,030	223,256	0.90
Fix-Few-Shot	GPT-3.5-t	133,903	98,260,778,555	313,466	0.81
Dyn.-Few-Shot	GPT-3.5-t	65,557	22,055,967,680	148,513	0.96
Zero-Shot	GPT-4-p	86,125	27,711,918,630	166,469	0.95
Fix-Few-Shot	GPT-4-p	95,265	43,749,594,680	209,164	0.92
Dyn.-Few-Shot	GPT-4-p	60,372	12,884,017,255	113,508	0.98
ML Solution:					
-	Polyn. Feature	307,997	308,327,631,192	555,273	0.41
-	ElasticNet	265,176	168,837,254,230	410,898	0.68
-	Linear	205091	102,227,548,423	319,730	0.80
-	Dec. Tree (D:4)	168,002	76,883,663,697	277,279	0.85
-	Ridge	98,139	54,038,016,561	232,460	0.90
-	Lasso	90,326	52,555,475,455	229,250	0.90
-	Dec. Tree	80,136	37,885,241,859	194,641	0.93
-	Random Forest	66,524	14,927,410,816	122,178	0.97

Table 4.8: Baseline Summary Price Prediction

Table 4.3 shows the entry count in the different datasets that have been derived from the single original dataset from Kaggle. The Price Average (Avg.) and Price

Standard deviation (Price Std.) are both given in INR. The baseline summary table 4.3.1, shows the LLM results for the Zero Shot, Fix Few Shot and Dynamic (Dyn.) Few Shot prompts on the GPT-3.5-turbo-0125 (GPT-3.5-t) and the GPT-4-0125-preview (GPT-4-p) models. It also shows the ML Results for different regression models. It evaluates on Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 score. In addition, Polynomial Feature Regression is represented by Polyn. Feature and Decision Tree is represented by Dec. Tree. The (D:4) following the Decision Tree indicates that this variant of the Decision Tree has a max_depth parameter of 4.

LLM Results

The performance of the GPT-3.5-t and GPT-4-p models varies significantly depending on the prompts used. The Dynamic Few Shot prompt with GPT-4-p achieves the best performance, evidenced by the lowest MAE, MSE, and RMSE. This suggests that GPT-4-p is better able to interpret and respond to prompts when provided with a dynamic few-shot learning context, resulting in significantly improved performance. Both GPT-3.5-t and GPT-4-p performed worse with the Fix-Few-Shot prompts compared to other prompt types. This could be due to the fix examples provided in the prompts not accurately representing the price distribution of the dataset. The Fix Few Shot examples have prices of 300,000 INR, 500,000 INR, and 375,000 INR. As a result, the average of the three randomly sampled examples provided in the few shot prompt is 391,667 INR, while the average of the test set is 624,817 INR (Table 4.3). As the model is provided with examples that are all significantly smaller than the average of the prices it is asked to predict, it is likely to structurally underestimate the prices. This raises an interesting question: will the LLM be able to identify this potential error class from the examples that will be provided to it later?

Regarding the Dyn.-Few-Shot prompt, the strong results can be attributed to the fact that when using a KNN to search for similar examples in the training set, cars with similar features are found, resulting in prices that are likely to be relatively close to the price to be predicted. Therefore, these examples guide the LMM to the correct price. This supports the theory that the LLM is influenced by the prices given in the examples and is more likely to choose a price within this range. Overall, it can be concluded from the table that the GPT-4-p model outperformed the GPT-3.5-t model for each prompt individually in all metrics. The performance increase in RMSE from GPT-3.5-t to GPT-4-p is relatively similar for the Zero Shot and Dyn. Few Shot prompts at 25% and 24%, respectively. However, the performance increase in RMSE for Fix Few Shot is significantly higher at 33%. Therefore, it seems that GPT-4-p relies more on its training data and the provided features, rather than being significantly influenced by the provided example prices, in comparison to GPT-3.5-t.

It is worth noting that GPT-3.5-t also had some problems with correct responses, with price predictions occasionally taking on a format like this: '5,50,000 INR'. Here, the LLM returned the price with an extra comma that does not make sense. The commas were removed before saving the responses. Due to the black-box nature of the LLM, it is unclear whether the model considered all variables or relied solely on prior knowledge of car names and their respective prices from the historical data it was trained on to produce the results. However, it is probable that the latter had a greater influence. The importance scores in the error analysis may offer further insights into this matter.

ML Solution

The Random Forest model demonstrates the best performance with the lowest MAE, MSE, and RMSE, alongside a high R^2 score of 0.97. This suggests that ensemble methods, particularly Random Forest, are highly effective in capturing the complex, non-linear relationships inherent in this dataset. The model's ability to aggregate decisions from a multitude of decision trees helps in reducing overfitting while maintaining the capacity to learn detailed patterns. On the contrary, Polynomial Feature Expansion exhibits the poorest performance, with the highest error metrics and a significantly lower R^2 score of 0.41. This could indicate that the approach overly complicates the model, introducing a high variance error due to overfitting, especially when the polynomial degrees are not carefully chosen or when regularization techniques are not applied adequately. It has to be noted that no fine-tuning was applied here. Models like ElasticNet, which combines the properties of both Ridge and Lasso regression, show a moderate performance, underscoring the value of regularization in balancing the bias-variance trade-off.

The performance disparity between Ridge and Lasso regression, both achieving a R^2 score of 0.90, yet with slightly different error metrics, points to the nuances in how each model handles feature selection and regularization. Lasso's ability to perform feature selection might have provided it a slight edge in handling irrelevant or less important features, although both models perform commendably well, indicating the dataset's linear components are significant. When Fine-Tuning the ElasticNet Model, it also showed a bias towards the Lasso Regression.

The Decision Tree was the second-best performing ML model. It was considered to provide the LLM with an explanation in the error analysis section by parsing the tree form of the decision tree. However, the full Decision Tree has a depth greater than 10, making it too large to parse into a representable form. Therefore, it was tested to limit the depth of the decision tree to a representable format. However, when applying a maximum depth of 4, which is a representable format, Lasso and Ridge Regression outperformed the decision tree in all parameters. Therefore, Lasso Regression was chosen for further error analysis due to its explainable weights and satisfactory performance.

The comparative analysis underscores the importance of model selection based on the characteristics of the data and the prediction task. While ensemble methods like Random Forest excel in handling complex datasets with high dimensional spaces and non-linear relationships, linear models retain their relevance, provided that appropriate regularization techniques are employed. Furthermore, more fine-tuning on the validation set could have potentially improved the models' performance further.

Costs

The cost of this analysis was \$0.04 per run for the GPT-3.5-t prompts and \$0.72 per run for the GPT-4-p prompts. The costs remained comparatively low due to the small number of output tokens for this task, as the model only had to respond with the task and nothing else. There were no costs for the ML models as they were run on locally.

Comparison & Discussion

When directly comparing the top performers, the Dynamic Few-Shot model of GPT-4-p and the Random Forest model, it is apparent that both achieve remarkably high R^2 scores (0.98 and 0.97, respectively), indicating strong predictive power. However, the LLM approach slightly edges out with lower error metrics (MAE, MSE, RMSE), suggesting a marginally better fit for the price prediction task at hand. The LLM's ability to leverage contextual understanding and natural language processing supposedly allows it to infer more nuanced patterns in the data, beyond what is captured by traditional features used in ML models. However, it is most likely that the LLM also has an edge in predicting due to its training on historical data that includes the cars from the dataset and their respective prices. Still, it is worth noting that GPT-4-p performs that well on a task that is typically better suited for regression models than for LLMs. This suggests that further research could be beneficial. It would be interesting to test this on a broader range of tasks and to fine-tune the ML models more rigorously.

4.3.2 Error Analysis

Four runs have been selected for the following steps in the price prediction error analysis. The first three runs were LLM runs conducted with GPT-4-p using the Zero-Shot, Fix-Few-Shot, and Dynamic-Few-Shot prompts. The fourth run was a Lasso Regression run will be compared with the other three. For this task it is not obvious what a false prediction and what a correct prediction is. In this thesis, a false prediction is classified as a prediction that deviates $\pm 20\%$ from the actual price. This information was given to the model in all following cases.

Confidence scores

The GPT-3.5-t model was used for the confidence scores due to cost considerations and the fact that this model was also used for the confidence scores of the other two tasks and comparability should be ensured. Due to the numerical structure of the predictions and actuals, no ROC curve was generated. Instead, the confidence metrics for the Zero-Shot, Fix-Few-Shot and Dyn.-Few-Shot runs are shown in table 4.9. Here the mean and standard deviation (Std) are given for correct predictions (less than $\pm 20\%$ deviation), incorrect predictions (more than $\pm 20\%$ deviation) and the overall mean and Std. It can be concluded that the confidence seems to be slightly higher in correct prediction cases than in incorrect ones, indicating that the model is able to accurately identify correct cases where it is 'more sure' of its prediction decision and other cases where it is 'less sure'. The differences in the confidence means appear to be small, but the overall low standard deviation must be taken into account, showing that the model only responded with values in a limited range of values. Several different prompts have been tried, but all result in the same small range of responses and resulting low standard deviation.

Prompt	Error	Mean	Std
Zero-Shot	Correct	85.69	2.24
Zero-Shot	Error	84.51	3.19
Zero-Shot	Overall	85.39	2.56
Fix-Few-Shot	Correct	83.41	3.07
Fix-Few-Shot	Error	82.64	3.01
Fix-Few-Shot	Overall	83.20	3.07
Dyn.-Few-Shot	Correct	84.12	4.51
Dyn.-Few-Shot	Error	81.06	4.06
Dyn.-Few-Shot	Overall	83.36	4.59

Table 4.9: Confidence Analysis Price Prediction

Unstructured Explanations

The unstructured explanations were carried out using GPT-4-p, as this was the same model used for the explanations in the other task. The responses for the unstructured explanations can be found in the code appendix B in the folder Datasets/Evaluations/Regression/Error_Analysis/structured_unstructured. The unstructured explanations often consist of a few sentences mentioning the attributes that the LLM found relevant to it's decision. However, the information provided does not go very deep or offer any new insights into a possible thought process. The response often looked like this: *"I made the prediction based on the year, mileage, fuel type, transmission, ownership history and engine size of the car. These attributes are crucial in determining the depreciation and market value of a used car."*.

Structured Explanations

Attribute	**Count**	**Mean**	**Std.**
km_driven	400	0.228	0.049
year	400	0.167	0.036
fuel	389	0.145	0.047
transmission	274	0.129	0.035
engine	390	0.127	0.029
owner	128	0.126	0.044
mileage	389	0.125	0.033
max_power	321	0.089	0.038
seller_type	33	0.083	0.030
torque	47	0.069	0.030
seats	20	0.060	0.026
name	0	0.000	0.000

Table 4.10: Structured Explanation Overview - Fix-Few-Shot - Price Prediction

In the structured explanations, the LLM was asked to list the attributes it considered relevant in a JSON format and assign an importance score. For all three runs, the most common attributes are km_driven, year and fuel. km_driven also has the highest average importance score. The results for the Fix-Few-Shot run can be seen in A.5 and the other two tables can be found in the appendix. The Zero-Shot run table can be found in the Appendix A.4 and the Dynamic-Few-Shot run can also be found in the Appendix A.6. What is interesting here is that the 'name' attribute is not really mentioned in the Structured Explanations, or only a few times. This is contrary to what one would think, as the car model name seems to be the most influential factor for an LLM to price a car, as it can use its prior training data to make better predictions. However, this does not seem to be a revelation about the inner workings of the LLM, but rather a basic error that indicates a lack of understanding by the LLM of its own predictions. This is due to the black-box factor, which does not allow the LLM to really 'explain' its own decision, but only to guess at the reasons for its own decision.

Error Classes

The error classes have been generated using GPT-4-p for the unstructured explanations, the structured explanations and the lasso regression weights as explanations. The error classes can all be found in Appendix A.3. The unstructured explanations can be found in table A.8, the error classes for the structured explanations can be found in table A.7 and the error classes for the lasso regression can be found in table A.9. The tables provide an overview of the error classes, but the LLM often provided further explanation of the error classes, and sometimes why the LLM assigned a particular error to a particular error class. The full error class responses can also be found in Appendix A.3.

For all runs the LLM responded with a minimum of four and a maximum of up to eight error classes. Overall, the error classes do not appear to be generic, but rather to list specific reasons for errors in the runs. There are also some similarities between the error classes of the different runs. For example, for the structured explanations, high mileage overestimation appears as an error class for each run, as well as some other error classes. The classes for all runs seem to be mostly disjoint, and for most runs, all the errors are sorted into the error classes. For some, where no number of occurrences is listed, the LLM replied that the errors for that error class are not really clear, but that there are probably several. However, there does not seem to be a pattern of error classes getting fewer occurrences or even error classes disappearing from zero-shot to fix-few-shot for the structured explanations. Notably, the dynamic-few-shot run has far fewer occurrences for each error class than the zero-shot run for the structured explanations, which seems to indicate that these error types were made less often for the dynamic-few-shot run, and the LLM found this out. The same pattern is repeated for the unstructured explanations. However, it is not as obvious here. The Zero-Shot error class "Misjudgement of Vehicle Attribute" with 18 occurrences shrinks and does not reappear or only reappears in a different form with fewer occurrences for the Fix-Few-Shot and the Dynamic-Few-Shot runs. It should also be noted that the LLM includes error classes that specifically mention errors from the few-shot examples, such as the error class "Inaccurate Comparison with Example Outputs" from the Dynamic-Few-Shot run in the Unstructured Explanations table A.7 (in Appendix A.3). This indicates a high level of understanding of the errors by the LLM, which was not really seen for the other tasks. The Lasso Regression error classes are evenly distributed in occurrence and are not too generic. However, they do not really take into account that the prediction was made by a lasso regression, even though the LLM was informed of this. A lasso regression cannot really make an error such as ”misjudging market trends” and it is unclear how the LLM derived such error classes from the weight explanations provided.

4.4 Results and Comparison

Prompt	**Model**	**Acc.**	**P**	**R**	**F1**
Schema Matching: (Multi-Column)					
Zero-Shot	GPT-4-p	0.88	0.88	0.38	0.52
Fix-Few-Shot	GPT-4-p	0.90	0.87	0.49	0.63
Dyn.-Few-Shot	GPT-4-p	0.91	0.91	0.53	0.67
-	Log. Reg.	0.46	0.18	0.61	0.28
Sentiment Analysis: (Single-Term)					
Zero-Shot	GPT-4-p	0.59	0.61	0.59	0.57
Fix-Few-Shot	GPT-4-p	0.60	0.66	0.60	0.57
Dyn.-Few-Shot	GPT-4-p	0.58	0.62	0.58	0.56
-	Log. Reg.	0.65	0.65	0.65	0.64
Price Prediction:		**MAE**	**MSE**	**RMSE**	$\mathbf{R^2}$
Zero-Shot	GPT-4-p	86,125	27,711,918,630	166,469	0.95
Fix-Few-Shot	GPT-4-p	95,265	43,749,594,680	209,164	0.92
Dyn.-Few-Shot	GPT-4-p	60,372	12,884,017,255	113,508	0.98
-	Lasso	90,326	52,555,475,455	229,250	0.90

Table 4.11: Summary of Results Used for Error Analysis - Price Prediction

To provide a comprehensive overview of all runs used for the error analysis, a summary table 4.11 was created. Upon comparing the F1 values of the LLM runs for the Schema Matching and Sentiment Analysis tasks, it can be concluded that they are relatively close, with an average F1 value of 0.57 for Sentiment Analysis and 0.61 for Schema Matching. However, the logistic regression performed much better in the sentiment analysis task than in the schema matching task. It even outperformed the LLM runs in the sentiment analysis task. It is surprising that a LLM does not perform better than a classical machine learning model, such as logistic regression, in more complex sentiment analysis tasks, such as aspect-based sentiment analysis. One would expect that a LLM would have significant advantages in understanding complex sentence structures and subtle meanings. The LLM performs more on par with the logistic regression model when forced to predict the sentiments of all terms in one answer, thus better understanding the context. However, it should be noted that this text presents a single-term prompting approach and some multi-term prompting approaches performed better than the ML solution. Additionally, it can be concluded that GPT-4-p performs surprisingly well in the price prediction task, where one would expect LLMs to be significantly disadvantaged. However, the results indicate that the LLM has the ability to perform as well as the best performing regression model and outperforming the Lasso regression in a Dynamic-Few-Shot setting, due to its historical background knowledge and the usage of the provided example being a significant advantage.

In terms of error classes, it is worth noting that the sentiment analysis task has significantly fewer errors compared to the schema matching task. The Sentiment Analysis LLM runs have a maximum of three error classes and a minimum of two, with an average of 2.67. On the other hand, the LLM error class runs in the schema matching task have a minimum of four error classes, a maximum of six, and an average of 5.00 error classes. The error prompt's similarity in both tasks suggests that there are more diverse error sources for the Schema Matching task than for sentiment analysis. Additionally, there appear to be more domains for explaining specific predictions. In terms of Logistic Regression runs, the difference between Schema Matching and sentiment analysis is not significant, with 5 error classes for Sentiment Analysis and 6 for Schema Matching. As the structure of the explanations for the logistic regression runs is relatively similar, this suggests that the varying numbers of error classes may be significantly influenced by the provided explanations and their diversity. The price prediction task also appears to have similar issues, with 6 error classes provided for the lasso regression and an average of 6.33 error classes per run for the LLM runs. This suggests that there are multiple sources of explanation and errors in the price prediction task, which may be attributed to the number of different attributes provided. However, it should be noted that the diversity in error classes for the price prediction task would not be as significant if the prompt did not prohibit answering with only 'overestimation' and 'underestimation' as error classes.

Chapter 5

Conclusion

5.1 Summary

The research aimed to determine the effectiveness of LLMs with different prompting strategies in solving the three tasks and how they compare to conventional machine learning models. The results indicate that the single column prompting approach was ineffective for the Schema Matching task, while the multi-column prompting approach, particularly when using the GPT-4-p model, performed significantly better. In conclusion, LLMs can be effective in solving Schema Matching tasks with the appropriate prompting methods and models. However, Logistic regression does not perform well in this task, with an F1 value of 0.28, which is lower than the worst-performing multi-column run. The Sentiment Analysis task demonstrated that both prompting approaches, single-term and multi-term, can effectively solve the task. The multi-term approach slightly outperforms the single-term approach in most settings when considering the weighted F1 values. Additionally, the logistic regression model achieved a surprisingly good F1 value, on par with the LLM performance. The Price Prediction analysis demonstrated that LLMs perform remarkably well, particularly when using the GPT-4-p models in a few-shot setting. Additionally, there is a performance disparity between fixed and dynamic few-shot prompt runs, with the dynamic runs outperforming the fixed ones. This suggests that providing more similar examples to the initial prompt can have an impact on performance. Furthermore, the LLM run that performed the best (GPT-4-p, Dynamic Few-Shot) outperformed even the top-performing regression run.

The second question addressed the extent to which LLMs can automatically analyse errors produced by both themselves and conventional machine learning models. In the Schema Matching task, the LLM performed satisfactorily in creating error classes and clustering errors for all three prompt types. It was also observed that the LLM produced different results in the structure and occurrence count of error classes for prompt types with different initial F1 value performance. Regarding the error classes generated by the logistic regression model, the results were not convincing as the LLM only produced general error classes without considering that the predictions were made using a machine learning model. However, for the Sentiment Analysis runs, the LLMs generated more generic and less varied error classes. The logistic regression model generated a good variety of disjunct error classes. Additionally, there was no significant difference in the error class structure between the different runs. For the Price Prediction task, all runs generated a good variety of exhaustive error classes, and the LLms demonstrated a good level of understanding of the errors and provided explanations.

Generally, it can be stated that GPT-4-p can be inconsistent in structuring its answers in regard to error classes, sometimes not providing error examples or counts. However, in most cases, the error classes were disjoint and no correct examples were falsely classified as errors. LLMs can generally solve all three task types satisfactorily and can also be used to automatically analyze and classify errors made by LLMs and classical machine learning methods, providing more insights into the decision-making process. However, their accuracy, insight potential and usefulness for further optimization in this topic remain uncertain and may vary for different tasks.

5.2 Future Work

The datasets used for the different tasks in this thesis were relatively small due to financial and computational limitations. It would be worthwhile to investigate whether the results would differ with larger or more diverse datasets. Additionally, the number of error examples provided to the LLM for error classification was limited. Testing a larger number of error cases, particularly with the release of LLMs with larger context windows, would be valuable. Additionally, it would be beneficial to evaluate the performance of GPT-3.5-t and other LLMs in automatic error classification, as in this thesis only GPT-4-p was used. This thesis ends after creating error classes and evaluating them qualitatively. Testing whether optimising LLMs that solve the original tasks using error class generations can improve performance would be an interesting topic of future work. Another interesting area for future work is to solve a wider range of quantitative regression tasks with LLMs to test the limits of LLMs in this task field. The initial results of subsection 4.3.2 of this thesis suggest that an LLM's prior knowledge of these tasks could potentially eliminate the need for large training sets, while still performing on par with classical regression models.

Bibliography

[1] Logistic regression for machine learning [a beginners guide]. 2024. https://www.knowledgehut.com/blog/data-science/logistic-regression-for-machine-learning, 2019.

[2] Shivang Agarwal, C. Ravindranath Chowdary, and Vivek Sourabh. Eazy learning: An adaptive variant of ensemble learning for fingerprint liveness detection, 2021.

[3] Kyle Aitken, Vinay V. Ramasesh, Yuan Cao, and Niru Maheswaranathan. Understanding how encoder-decoder architectures attend, 2021.

[4] Serkan Akinci, Erdener Kaynak, Eda Atilgan, and Safak Aksoy. Where does the logistic regression analysis stand in marketing literature?: A comparison of the market positioning of prominent marketing journals. *European Journal of Marketing*, 41:537–567, 06 2007.

[5] Ali Al-Aradi. Credit scoring via logistic regression. *Expert Systems with Applications*, 2014.

[6] Arnau Albà, Andreas Adelmann, Lucas Münster, Dimitri Rochman, and Romana Boiger. Fast uncertainty quantification of spent nuclear fuel with neural networks, 2023.

[7] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of open language models, 2023.

[8] S. Amarendra. Optimizing parameter efficiency in machine learning models: A focus on reducing memory overhead with l-bfgs-optimized algorithms. *Journal of Electrical Systems*, 20:298–314, 04 2024.

[9] Simran Arora, Avanika Narayan, Mayee F. Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. Ask me anything: A simple strategy for prompting language models, 2022.

[10] Robert L. Logan IV au2, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models, 2021.

[11] Shafna Fitria Nur Azizah, Hasan Dwi Cahyono, Sari Widya Sihwi, and Wisnu Widiarto. Performance analysis of transformer based models (BERT, ALBERT and RoBERTa) in fake news detection, 2023.

[12] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):1001–1008, 1989.

[13] Protima Banerjee and Hyoil Han. Language modeling approaches to information retrieval. *JCSE*, 3:143–164, 09 2009.

[14] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, nov 2020. Association for Computational Linguistics.

[15] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. 2003.

[16] Leo Breiman. Random forests. 45(1):5–32, 2001.

[17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Krueger, et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[18] Paul Burkhardt. Evaluation of chain-of-thought prompting on the task of entity matching, Bachelor thesis, University of Mannheim, 2023.

[19] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in large language models: a comprehensive review, 2023.

[20] Alexander Clark, Gianluca Giorgolo, and Shalom Lappin. Statistical representation of grammaticality judgements: the limits of n-gram models. In Vera Demberg and Roger Levy, editors, *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 28–36. Association for Computational Linguistics, 2013.

[21] Reng Chiz Der and Christian Bizer. WDC schema matching benchmark (SMB), 2024. https://webdatacommons.org/structureddata/smb/.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding, 2019.

[23] Hong-Hai Do and Erhard Rahm. COMA — a system for flexible combination of schema matching approaches. In *VLDB '02: Proceedings of the 28th International Conference on Very Large Databases*, pages 610–621. Elsevier, 2002.

[24] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the semantic web. 12(4):303–319, 2003.

[25] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023.

[26] Matthias Fey, Jan E. Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus, 2020.

[27] Paul Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation*. 05 2017.

[28] Jianfeng Gao and Chin-Yew Lin. Introduction to the special issue on statistical language modeling. 3(2):87–93, 2004.

[29] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. KNN model-based approach in classification. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 986–996. Springer, 2003.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[31] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.

[32] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Hennigan, et al. Training compute-optimal large language models, 2022.

[33] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020.

[34] Hyun il Lim. A linear regression approach to modeling software characteristics for classifying similar software. *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, 1:942–943, 2019.

[35] Taghreed M. Jawa. Logistic regression analysis for studying the impact of home quarantine on psychological health during COVID-19 in saudi arabia. 61(10):7995–8005, 2022.

[36] Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. A challenge dataset and effective models for aspect-based sentiment analysis. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6280–6285. Association for Computational Linguistics, 2019.

[37] Chuyang Jin. Price prediction of used cars using machine learning. In *2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT)*, pages 223–230, 2021.

[38] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models, 2023.

[39] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[40] Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget. Recurrent neural network based language modeling in meeting recognition. In *Interspeech 2011*, pages 2877–2880. ISCA, 2011.

[41] Christos Koutras, Marios Fragkoulis, Asterios Katsifodimos, and Christoph Lofi. REMA: Graph embeddings-based relational schema matching. 2020.

[42] Sudhanshu Kumar, Partha Pratim Roy, Debi Prosad Dogra, and Byung-Gyu Kim. A comprehensive review on sentiment analysis: Tasks, approaches and applications, 2023.

[43] Ivano Lauriola, Alberto Lavelli, and Fabio Aiolli. An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, 470:443–456, 2022.

[44] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. Table-gpt: Table-tuned gpt for diverse table tasks, 2023.

[45] Wen-Syan Li and Chris Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. 33(1):49–84, 2000.

[46] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022.

[47] Octavio Loyola-González. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access*, 7:154096–154113, 10 2019.

[48] Qingyu Lu, Liang Ding, Liping Xie, Kanjian Zhang, Derek F. Wong, and Dacheng Tao. Toward Human-Like Evaluation for Natural Language Generation with Error Analysis, 2022.

[49] Qingyu Lu, Baopu Qiu, Liang Ding, Kanjian Zhang, Tom Kocmi, and Dacheng Tao. Error analysis prompting enables human-like translation evaluation in large language models, 2024.

[50] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity, 2022.

[51] Huan Ma, Changqing Zhang, Yatao Bian, Lemao Liu, Zhirui Zhang, Peilin Zhao, Shu Zhang, Huazhu Fu, Qinghua Hu, and Bingzhe Wu. Fairness-guided few-shot prompting for large language models, 2023.

[52] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. Generic schema matching with cupid. *Proc 27th VLDB Conference*, 07 2001.

[53] Dastan Maulud and Adnan Mohsin Abdulazeez. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, 1:140–147, 12 2020.

[54] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[55] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech 2010*, pages 1045–1048. ISCA, 2010.

[56] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. 1998.

[57] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 412–418. Association for Computational Linguistics, 2004.

[58] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5):183–197, 1991.

[59] Michelangelo Naim, Mikhail Katkov, Sandro Romani, and Misha Tsodyks. Fundamental law of memory recall. *Physical Review Letters*, 124(1), January 2020.

[60] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2024.

[61] Feng Nie, Meixi Chen, Zhirui Zhang, and Xu Cheng. Improving few-shot performance of language models via nearest neighbor calibration, 2022.

[62] OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, et al. Gpt-4 technical report, 2024.

[63] Ralph Peeters and Christian Bizer. Using chatgpt for entity matching, 2023.

[64] Joanne Peng, Kuk Lee, and Gary Ingersoll. An introduction to logistic regression analysis and reporting. *Journal of Educational Research - J EDUC RES*, 96:3–14, 09 2002.

[65] Vagelis Plevris, German Solorzano, Nikolaos Bakas, and Mohamed Ben Seghier. Investigation of performance metrics in regression analysis and machine learning-based prediction models. 06 2022.

[66] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 task 4: Aspect based sentiment analysis. In Preslav Nakov and Torsten Zesch, editors, *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35. Association for Computational Linguistics, 2014.

[67] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8, 07 2009.

[68] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, and Rada Mihalcea. Beneath the tip of the iceberg: Current challenges and new directions in sentiment analysis research, 2020.

[69] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[70] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. 10(4):334–350, 2001.

[71] Hassan Ramchoun, Mohammed Amine, Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. Multilayer perceptron: Architecture optimization and training. 4(1):26, 2016.

[72] Matthew Renze and Erhan Guven. The effect of sampling temperature on problem solving in large language models, 2024. version: 1.

[73] Lior Rokach and Oded Maimon. *Decision Trees*, volume 6, pages 165–192. 01 2005.

[74] H. Roopa and T. Asha. A linear model based on principal component analysis for disease prediction. *IEEE Access*, 7:105314–105318, 2019.

[75] R. Rosenfeld. Two decades of statistical language modeling: where do we go from here? 88(8):1270–1278, 2000.

[76] Kavitha S, Varuna S, and Ramya R. A comparative analysis on linear regression and support vector regression. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–5, 2016.

[77] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

[78] Robin M. Schmidt. Recurrent neural networks (RNNs): A gentle introduction and overview, 2019.

[79] Eitam Sheetrit, Menachem Brief, Moshik Mishaeli, and Oren Elisha. Rematch: Retrieval enhanced schema matching with llms, 2024.

[80] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. 25(1):158–176, 2013.

[81] Derya Soydaner. Attention mechanism in neural networks: where it comes and where it goes. 34(16):13371–13385, 2022.

[82] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[83] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, et al. Gemini: A family of highly capable multimodal models, 2023.

[84] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[85] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[86] Fouad Trad and Ali Chehab. Evaluating the efficacy of prompt-engineered large multimodal models versus fine-tuned vision transformers in image-based security applications, 2024.

[87] Peter Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the*

Association for Computational Linguistics, pages 417–424. Association for Computational Linguistics, 2002.

[88] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. A review on the long short-term memory model. *Artificial Intelligence Review*, 53, 12 2020.

[89] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[90] Colin Kai Wang. Sentiment analysis using support vector machines, neural networks, and random forests. In Pushpendu Kar, Jiayang Li, and Yuhang Qiu, editors, *Proceedings of the 2023 International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2023)*, volume 108, pages 23–34. Atlantis Press International BV, 2023. Series Title: Advances in Computer Science Research.

[91] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[92] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[93] BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model, 2023.

[94] Skyler Wu, Eric Meng Shen, Charumathi Badrinath, Jiaqi Ma, and Himabindu Lakkaraju. Analyzing chain-of-thought prompting in large language models via gradient-based feature attributions, 2023.

[95] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2023.

[96] Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models, 2023.

[97] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. 8(4):e1253, 2018.

[98] Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Jialin Pan, and Lidong Bing. Sentiment analysis in the era of large language models: A reality check, 2023.

[99] Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. A survey on aspect-based sentiment analysis: Tasks, methods, and challenges, 2022.

[100] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023.

[101] Mingqian Zheng, Jiaxin Pei, and David Jurgens. Is "a helpful assistant" the best role for large language models? a systematic evaluation of social roles in system prompts, 2023.

[102] Mingyu Zong and Bhaskar Krishnamachari. a survey on gpt-3, 2022.

[103] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix A

Further Experimental Results

A.1 Schema Matching

Error Analysis

Error Classes - Zero-Shot - Unstructured

```
Based on the provided matching decisions and explanations, the
    wrong decisions can be grouped into distinct fault categories
    as follows:

1. **Mismatch in Data Type or Format**:
   - Occurrences: 5
   - Examples: Matching decisions where numerical values were
       incorrectly matched with textual data, or specific formats
       (e.g., dates, alphanumeric strings) did not align between
       tables.

2. **Incorrect Contextual Matching**:
   - Occurrences: 7
   - Examples: Decisions where the context or semantic meaning of
       the columns was misunderstood, leading to incorrect matches
        (e.g., matching company names with numerical IDs,
       geographical names with financial figures).

3. **Misinterpretation of Content**:
   - Occurrences: 3
   - Examples: Cases where the content of the columns was
       misinterpreted, leading to incorrect conclusions about the
       match (e.g., interpreting elevation data as unrelated
       numerical values).

4. **Overlooking Direct Correspondence**:
```

```
   - Occurrences: 5
   - Examples: Instances where there was a direct correspondence
       between columns in terms of data type, format, and context,
        but the decision was made not to match them due to
       oversight or misunderstanding of the data.

These categories encompass the primary types of errors observed in
     the provided matching decisions, highlighting challenges in
    data type/format recognition, contextual understanding,
    content interpretation, and identifying direct correspondences
     between datasets.
```

Error Classes - Fix-Few-Shot - Unstructured

```
Based on the provided examples and decisions, the wrong decisions
    can be grouped into distinct fault categories as follows:

1. **Mismatch in Data Type or Context**:
   - This category includes decisions where the matching was
       incorrect due to a fundamental difference in the type of
       data (e.g., numerical vs. textual) or the context of the
       data (e.g., geographical names vs. service features).
   - Occurrences: Multiple instances across tasks, such as
       matching game titles with publishers, matching country
       names with specific legislative bodies, and matching
       geographical features with unrelated categories.

2. **Incorrect Interpretation of Data Content**:
   - Decisions that were incorrect due to a misunderstanding or
       misinterpretation of the content within the columns. For
       example, matching a column of descriptive text with a
       column of publisher names, or matching a column of service
       features with a column of company names.
   - Occurrences: Several instances, particularly in tasks
       involving matching game titles with publishers or
       developers, and matching geographical descriptions with
       specific locations or entities.

3. **Overlooking Specificity or Granularity of Data**:
   - Incorrect matches that occurred because the decision failed
       to account for the level of specificity or granularity
       between the columns. For example, matching broad
       geographical regions with specific mountain names or
       matching general service descriptions with specific airline
        names.
   - Occurrences: Notable in tasks involving geographical data,
       such as matching mountain heights with geographical
```

```
      locations, and matching airline services with specific
      airline names.

4. **Failure to Recognize Unique Identifiers or Codes**:
   - Incorrect decisions where unique identifiers or codes in one
      column were not correctly matched with their counterparts
      or related data in another column. This includes failing to
       match abbreviations or codes representing regions with
      their detailed descriptions or failing to match specific
      identifiers with entities.
   - Occurrences: Evident in tasks involving matching
      abbreviations or codes with detailed geographical locations
       or organizational entities.

5. **Ignoring Temporal Data Mismatch**:
   - Wrong decisions that occurred due to not recognizing
      mismatches in temporal data, such as matching columns
      containing different formats of dates or matching columns
      with temporal data against columns with non-temporal data.
   - Occurrences: Seen in tasks where release dates or historical
      dates were involved, and the matching failed to account for
       the difference in temporal data presentation.

6. **General Misalignment of Categories**:
   - Incorrect matches due to a general misalignment of categories
       between the columns, where the columns represented
      fundamentally different categories or types of information
      that were not related or comparable.
   - Occurrences: Present in various tasks, especially where one
      column represented a category or type of information that
      had no direct relation or comparability to the category or
      type of information in the matched column.

These fault categories highlight the challenges in accurately
    interpreting and matching data across different contexts,
    formats, and levels of specificity or granularity.
```

Error Classes - Dynamic-Few-Shot - Unstructured

```
Fault Categories and Occurrences:

1. **Mismatch in Data Type and Context**:
   - **Column A-3 and Column B-5 do not match**: This decision was
       made 4 times. The mismatch was due to one column
      containing geographical locations or countries and the
      other containing specific addresses or locations related to
       companies, showing a fundamental difference in the level
      of geographical detail.
```

```
- **Column A-3 and Column B-2 do not match**: This decision was
    made 3 times. The mismatch was due to one column
   containing years and the other containing full dates (year,
    month, day), indicating a discrepancy in data format and
   specificity.
- **Column A-4 and Column B-3 do not match**: This decision was
    made 2 times. The mismatch was due to one column listing
   video game titles and the other describing the type of
   video game, showing a fundamental difference in the type of
    information each column represents.
- **Column A-4 and Column B-2 do not match**: This decision was
    made 1 time. The mismatch was due to one column containing
    genres of video games and the other containing dates or
   being left blank, indicating a fundamental difference in
   data type and context.

2. **Incorrect Matching Based on Professional Context**:
   - **Column A-1 and Column B-1 match**: This decision was made 1
       time. The mismatch was due to incorrectly matching columns
       based on the observation that both columns contain names
      of individuals associated with the creation or production
      of films, without considering the exact match of names.

3. **Incorrect Matching Based on Specific Titles vs. Gameplay
    Modes**:
   - **Column A-0 and Column B-0 do not match**: This decision was
       made 1 time. The mismatch was due to one column listing
      specific game titles and the other listing different sets
      of entities within the airline industry, with no direct one
      -to-one correspondence between the specific names listed in
       each table.

4. **Mismatch in Specificity and Context of Geographical
    Information**:
   - **Column A-3 and Column B-0 do not match**: This decision was
       made 1 time. The mismatch was due to one column containing
       broad, country-level location information and the other
      offering specific cities and sometimes even addresses,
      indicating a discrepancy in the granularity of location
      data.

5. **Mismatch in Data Representation (Categorical vs. Date-Based)
    **:
   - **Column A-1 and Column B-3 do not match**: This decision was
       made 1 time. The mismatch was due to one column being
      categorical, describing the type or genre of video games,
      while the other is primarily date-based, indicating release
       dates or is otherwise unrelated to game genre.
```

In summary, the most common fault category is the mismatch in data type and context, occurring 10 times across different examples. The incorrect matching based on professional context and specific titles vs. gameplay modes each occurred once, while the mismatch in specificity and context of geographical information and mismatch in data representation also occurred once each.

Error Classes - Logistic Regression

Based on the provided examples, the wrong decisions made by the Logistic Regression Machine Learning Model can be grouped into distinct fault categories as follows:

1. **Mismatched Content Types**:
   - This category includes decisions where the model incorrectly matched columns that contain fundamentally different types of content, such as numeric values with text descriptions or identifiers with addresses.
   - Occurrences: 7 (Tasks 1, 2, 3, 4, 5, 6, 7)

2. **Geographical Misinterpretations**:
   - Decisions that incorrectly matched columns based on geographical names or locations, without considering the context or specificity required for a correct match.
   - Occurrences: 3 (Tasks 8, 16, 20)

3. **Incorrect Matching of Identifiers or Codes**:
   - This category involves decisions where the model incorrectly matched columns containing identifiers, codes, or similar discrete data points, such as game titles with game genres or broadcasting frequencies with station names.
   - Occurrences: 4 (Tasks 10, 11, 12, 18)

4. **Temporal or Contextual Misalignment**:
   - Decisions where the model failed to correctly interpret the temporal or contextual relevance of the data, leading to incorrect matches between columns that are not logically or contextually aligned.
   - Occurrences: 2 (Tasks 13, 19)

5. **Misinterpretation of Numerical Data**:
   - Incorrect matches based on numerical data where the model failed to recognize the significance or context of the numbers, leading to matches between unrelated columns.
   - Occurrences: 2 (Tasks 17, 18)

6. **General Misunderstanding of Data Context**:

```
   - This category includes decisions where the model made
      incorrect matches due to a general misunderstanding of the
      data context, such as matching columns based on superficial
       similarities without understanding the underlying context.
   - Occurrences: 2 (Tasks 14, 15)

In summary, the most common fault category is "Mismatched Content
    Types," with 7 occurrences, followed by "Incorrect Matching of
     Identifiers or Codes" and "Geographical Misinterpretations."
    The least common categories are "Temporal or Contextual
    Misalignment," "Misinterpretation of Numerical Data," and "
    General Misunderstanding of Data Context," each with 2
    occurrences.
```

Error Class Overview

	Zero-Shot	**Fix-Few-Shot**	**Dyn.-Few-Shot**	**Log. Reg.**
Class 1	Mismatch in Data Type or Format (5)	Mismatch in Data Type or Context (-)	Mismatch in Data Type and Context (5)	Mismatched Content Types (7)
Class 2	Incorrect Contextual Matching (7)	Incorrect Interpretation of Data Content (-)	Incorrect Matching Based on Professional Context (1)	Geographical Misinterpretations (3)
Class 3	Misinterpretation of Content (3)	Overlooking Specificity or Granularity of Data (-)	Incorrect Matching Based on Specific Titles vs. Gameplay Modes (1)	Incorrect Matching of Identifiers or Codes (4)
Class 4	Overlooking Direct Correspondence (5)	Failure to Recognize Unique Identifiers or Codes (-)	Mismatch in Specificity and Context of Geographical Information (1)	Temporal or Contextual Misalignment (2)
Class 5	-	Ignoring Temporal Data Mismatch (-)	Mismatch in Data Representation (Categorical vs. Date-Based) (1)	Misinterpretation of Numerical Data (2)
Class 6	-	General Misalignment of Categories (-)	-	General Misunderstanding of Data Context (2)

Table A.1: Error Class Overview with # of occurrences - Schema Matching

A.2 Sentiment Analysis

Baseline Analysis

Prompt	Model	Acc.	P	R	F1	Confusion Matrix
Single Term:						
Zero-Shot	GPT-3.5-t	0.59	0.60	0.59	0.57	[77, 31, 9] [34, 71, 89] [6, 15, 119]
Zero-Shot	GPT-4-p	0.59	0.61	0.59	0.57	[96, 13, 8] [56, 64, 74] [14, 19, 107]
Fix-Few-Shot	GPT-3.5-t	0.48	0.51	0.48	0.40	[100, 7, 10] [96, 15, 83] [32, 6, 102]
Fix-Few-Shot	GPT-4-p	0.60	0.66	0.60	0.57	[100, 6, 11] [64, 59, 71] [21, 9, 110]
Dyn.-Few-Shot	GPT-3.5-t	0.63	0.63	0.63	0.63	[75, 34, 8] [35, 112, 47] [3, 38, 99]
Dyn.-Few-Shot	GPT-4-p	0.58	0.62	0.58	0.56	[95, 11, 11] [62, 61, 71] [20, 13, 107]

Table A.2: Baseline summary of weighted averages - Only Single Term - Sentiment Analysis

Prompt	**Model**	**Acc.**	**P**	**R**	**F1**	**Confusion Matrix**
Multi Term:						
Zero-Shot	GPT-3.5-t	0.64	0.66	0.64	0.62	[88, 18, 11] [32, 80, 82] [6, 15, 119]
Zero-Shot	GPT-4-p	0.67	0.69	0.67	0.67	[90, 18, 9] [28, 105, 51] [18, 13, 109]
Fix-Few-Shot	GPT-3.5-t	0.63	0.65	0.63	0.61	[92, 15, 10] [28, 81, 75] [14, 16, 110]
Fix-Few-Shot	GPT-4-p	0.65	0.67	0.65	0.64	[92, 15, 10] [44, 95, 55] [16, 18, 106]
Dyn.-Few-Shot	GPT-3.5-t	0.63	0.65	0.63	0.62	[88, 18, 11] [43, 87, 64] [12, 18, 110]
Dyn.-Few-Shot	GPT-4-p	0.67	0.67	0.67	0.66	[85, 22, 10] [41, 109, 44] [13, 21, 106]
ML Solution:						
-	Log. Reg.	0.65	0.65	0.65	0.64	[72, 27, 18] [12, 155, 27] [23, 51, 66]

Table A.3: Baseline summary of weighted averages - Multi Term & ML - Sentiment Analysis

Error Analysis

ROC Curves Sentiment Analysis

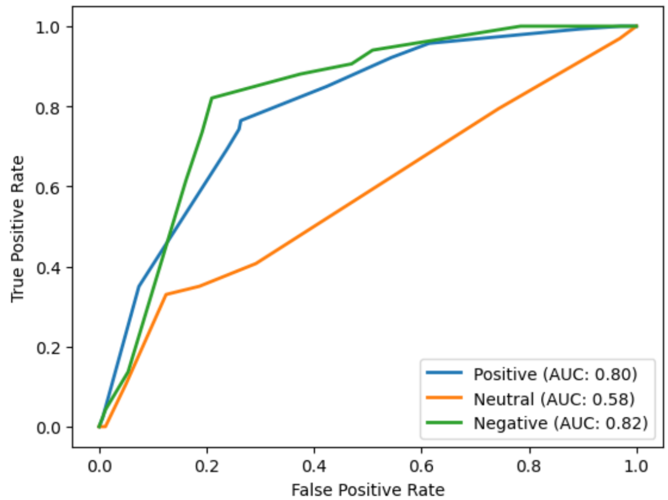

Figure A.1: ROC Zero-Shot run Sentiment Analysis

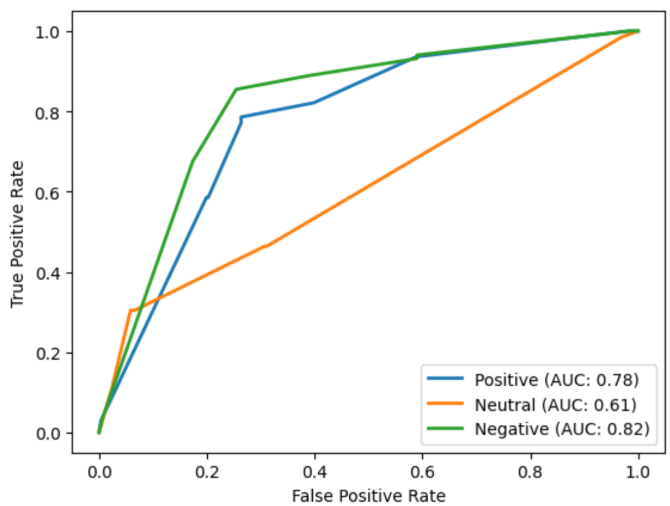

Figure A.2: ROC Curve Fix-Few-Shot run Sentiment Analysis

Error Classes - Zero-Shot - Unstructured

```
Based on the provided false decisions, we can categorize the
    errors into distinct fault categories as follows:

1. **Misinterpretation of Contextual Sentiment**:
   - This occurs when the sentiment towards a specific aspect is
       incorrectly inferred due to a misunderstanding of the
       overall context or the influence of nearby negative or
       positive words that do not directly relate to the aspect in
        question.
   - Occurrences: Task 1, Task 2, Task 3, Task 4, Task 5, Task 6,
       Task 7, Task 8, Task 9, Task 10, Task 11, Task 12, Task 18,
        Task 20

2. **Neutral Misclassification**:
   - This error type involves incorrectly classifying a neutral
       sentiment as either positive or negative (or vice versa)
       due to overemphasis on certain words or phrases that do not
        actually convey a strong sentiment towards the aspect.
   - Occurrences: Task 13 (positive to neutral), Task 16 (negative
        to neutral), Task 17 (neutral to negative), Task 19 (
       positive to negative)

3. **Overgeneralization of Sentiment**:
   - This fault occurs when the sentiment towards one aspect of a
       prompt is overgeneralized to another aspect, even when the
       sentiment towards the second aspect is not explicitly
       stated or is neutral.
   - Occurrences: Task 14, Task 15

Summary of Fault Categories and Frequencies:
- **Misinterpretation of Contextual Sentiment**: 14 occurrences
- **Neutral Misclassification**: 4 occurrences
- **Overgeneralization of Sentiment**: 2 occurrences

These categories highlight the challenges in accurately
    interpreting sentiment in aspect-based sentiment analysis,
    especially when dealing with nuanced or complex expressions of
     sentiment that require a deep understanding of context and
    the specific aspect being evaluated.
```

Error Classes - Zero-Shot - Structured

```
The false decisions can be grouped into distinct fault categories
    as follows:

1. **Misinterpretation of Contextual Sentiment**:
   - Task 1: The sentiment towards "stuff" was influenced by
       negative descriptors ("greasy", "stinky") but failed to
```

```
      recognize the overall positive sentiment of seeking a
      better alternative.
   - Task 13: The sentiment towards "dish" was neutral, failing to
       recognize the positive aspect of receiving something for
      free despite the critique of food quality.
   - Task 16: The sentiment towards "main course" was marked as
      negative based on the word "ok", not recognizing that "ok"
      in this context does not strongly denote negativity.
   - Task 17: The sentiment towards "setting" was marked as
      neutral, failing to recognize that being described in
      conjunction with "average service" implies a negative
      sentiment.
   - Occurrences: 4

2. **Incorrect Sentiment Attribution**:
   - Task 2: The sentiment towards "menu" was marked as negative
      due to price increase, which is an incorrect attribution as
       the sentiment towards the menu itself should remain
      neutral.
   - Task 4: The sentiment towards "tomato and onions" was marked
      as positive based on preference advice, which should not
      affect the neutral sentiment towards the ingredients
      themselves.
   - Task 5: The sentiment towards "appetizers" was marked as
      positive influenced by the service, which is an incorrect
      attribution.
   - Task 6: The sentiment towards "chocolate sauce tic-tac-toe"
      was marked as positive without a clear negative or positive
       aspect directly related to it.
   - Task 7: The sentiment towards "wine" was marked as positive
      influenced by service aspects, not the wine itself.
   - Task 8: The sentiment towards "dishes" was marked as negative
       due to being late, which affects the service, not the
      dishes themselves.
   - Task 9: The sentiment towards "plates" was marked as positive
       based on the food items, which should not directly
      influence the sentiment towards the plates themselves.
   - Task 10: The sentiment towards "white sangria" was marked as
      positive based on an assumption of being "refreshing",
      which is not directly stated.
   - Task 11: The sentiment towards "Catfish" was marked as
      positive influenced by the sauce, which is an incorrect
      attribution.
   - Task 12: The sentiment towards "Hawaiian rib" was marked as
      positive based on assumed qualities, not directly stated in
       the prompt.
   - Task 18: The sentiment towards "goat cheese" was marked as
      positive influenced by its pairing with "smoked salmon",
      which is an incorrect attribution.
```

```
   - Task 19: The sentiment towards "bar" was marked as positive
       based on hypothetical improvement to atmosphere, which is
       an incorrect attribution.
   - Task 20: The sentiment towards "dessert" was marked as
       positive based on the description of one option over
       another, which should not affect the overall sentiment
       towards "dessert" as a category.
   - Occurrences: 14

3. **Failure to Recognize Neutral Sentiment**:
   - Task 3: The sentiment towards "dessert" was marked as
       positive when it should have been neutral, as bringing a
       dessert does not inherently carry a positive or negative
       sentiment.
   - Task 14: The sentiment towards "bar" was marked as neutral,
       failing to recognize the negative sentiment implied by "
       harsh lighting".
   - Task 15: The sentiment towards "salmon" was marked as neutral
       , failing to recognize the negative sentiment implied by
       the inability to afford it.
   - Occurrences: 3

In summary, the false decisions can be categorized into three main
     fault categories with the following occurrences:
    Misinterpretation of Contextual Sentiment (4), Incorrect
    Sentiment Attribution (14), and Failure to Recognize Neutral
    Sentiment (3).
```

Error Classes - Fix-Few-Shot - Unstructured

```
The false decisions can be grouped into distinct fault categories
    as follows:

1. **Misinterpretation of Context or Detail**:
   - Task 1 (rice pudding): The negative sentiment was based on a
       comparison, but the actual sentiment was positive.
   - Task 2 (Food): Positive sentiment was inferred from "generous
        portions," ignoring the negative context of being
       expensive.
   - Task 3 (candle): Positive sentiment was inferred from the
       action of bringing a dessert with a candle, but the actual
       sentiment towards the candle itself was neutral.
   - Task 4 (pork chop): Negative sentiment was inferred from the
       description of accompanying sauces, but the actual
       sentiment towards the pork chop was neutral.
   - Task 5 (lounge): Negative sentiment was inferred from the
       lack of space, but the actual sentiment towards the lounge
       was neutral.
```

```
  - Task 6 (table): Negative sentiment was inferred from the
     waiting time, but the actual sentiment towards the table
     was neutral.
  - Task 7 (waiters): Positive sentiment was inferred from
     descriptive words, but the actual sentiment was negative.
  - Task 8 (beginning appetizers): Positive sentiment was
     inferred from the description of other items, but the
     actual sentiment towards the beginning appetizers was
     neutral.
  - Task 9 (dining): Positive sentiment was inferred from the
     description of the food, but the actual sentiment towards
     dining was neutral.
  - Task 10 (menu): Positive sentiment was inferred from the
     waiter's actions, but the actual sentiment towards the menu
      was neutral.
  - Task 11 (decor): Neutral sentiment was inferred from mixed
     descriptions, but the actual sentiment was positive.
  - Task 12 (edamame): Positive sentiment was inferred from
     receiving it for free, but the actual sentiment was neutral
     .
  - Task 13 (main courses): Negative sentiment was inferred from
     comparison with appetizers, but the actual sentiment was
     neutral.
  - Task 14 (menu): Negative sentiment was inferred from poor
     lighting affecting menu readability, but the actual
     sentiment was neutral.
  - Task 15 (fish): Negative sentiment was inferred from the
     action of having to "fish out" a bottle, but the actual
     sentiment was neutral.
  - Task 16 (appetites): Negative sentiment was inferred from the
      phrase "won't be enough," but the actual sentiment was
     positive.
  - Task 17 (lunch): Negative sentiment was inferred from the
     context of waiting for an average burger, but the actual
     sentiment towards lunch was neutral.
  - Task 18 (fries): Negative sentiment was inferred from a
     comparison, but the actual sentiment was positive.
  - Task 19 (shrimp dishes): Positive sentiment was inferred from
      the mention of loving shrimp roles, but the actual
     sentiment was neutral.
  - Task 20 (bagel): Positive sentiment was inferred from the
     suggestion to try it with spreads, but the actual sentiment
      was neutral.

Occurrences: 20

2. **Incorrect Sentiment Polarity**:
  - No clear cases where the sentiment polarity was simply
     reversed without context misinterpretation or detail
```

```
       oversight.

3. **Overgeneralization**:
   - Tasks where the sentiment towards a specific aspect was
       inferred based on the general sentiment of the sentence or
       another aspect, leading to incorrect predictions. This can
       be seen in tasks like Task 10 (menu), Task 14 (menu), and
       Task 15 (fish), where the sentiment towards the specific
       aspect was neutral, but the decision was influenced by the
       sentiment towards another aspect or the general situation.

Occurrences: Could be argued for several tasks, but the primary
    issue lies in misinterpretation or overlooking details.

In summary, the most common fault category is the **
    Misinterpretation of Context or Detail**, occurring in all 20
    false decisions. There's no clear case of Incorrect Sentiment
    Polarity as a distinct category, and Overgeneralization is a
    potential sub-issue within the misinterpretation category,
    affecting the accurate understanding of the sentiment towards
    specific aspects.
```

Error Classes - Fix-Few-Shot - Structured

```
Based on the provided false decisions, we can categorize the
    errors into distinct fault categories as follows:

1. **Misinterpretation of Context or Negation**:
   - Task 2 (Food): The sentiment was marked as positive due to "
       generous" portions, ignoring the negative context of not
       being for those on a budget.
   - Task 4 (Pork chop): The sentiment was marked as negative due
       to the descriptors "too sour" and "too sweet" for
       accompanying items, not directly related to the pork chop
       itself.
   - Task 10 (Menu): The sentiment was marked as positive based on
        the waiter's actions and recommendations, not directly
       reflecting the sentiment towards the menu itself.
   - Task 16 (Apitites): The sentiment was marked as negative due
       to "won't be enough," ignoring the positive aspect of "
       hearty appetites."
   - Task 17 (Lunch): The sentiment was marked as negative based
       on the context of waiting, not the lunch itself.
   - Task 18 (Fries): The sentiment was marked as negative based
       on the comparison with better options elsewhere, not
       directly on the fries.
   - Occurrences: 7
```

```
2. **Incorrect Sentiment Attribution**:
   - Task 1 (Rice pudding): The sentiment was marked as negative
     based on the comparison with flan, ignoring the positive
     aspects mentioned for flan.
   - Task 3 (Candle): The sentiment was marked as positive based
     on the overall positive actions of the staff, not the
     candle itself.
   - Task 5 (Lounge): The sentiment was marked as negative based
     on the lack of space, not the lounge area itself.
   - Task 6 (Table): The sentiment was marked as negative based on
      the wait time, not the table itself.
   - Task 7 (Waiters): The sentiment was marked as positive based
     on the friendly address, ignoring the negative context.
   - Task 8 (Beginning appetizers): The sentiment was marked as
     positive based on the quality of other dishes, not the
     appetizers themselves.
   - Task 9 (Dining): The sentiment was marked as positive based
     on the quality of dishes, ignoring the negative context of
     discontinuing lunch.
   - Task 11 (Decor): The sentiment was marked as neutral despite
     positive aspects highlighted by the packed crowds and
     critics' praises.
   - Task 13 (Edamame): The sentiment was marked as positive based
      on being free, ignoring the neutral context.
   - Task 14 (Main courses): The sentiment was marked as negative
     based on being compared to appetizers, not directly on
     their own quality.
   - Task 15 (Fish): The sentiment was marked as negative based on
      the effort to get the wine, not the fish itself.
   - Task 19 (Shrimp dishes): The sentiment was marked as positive
      based on other dishes, not directly on the shrimp dishes.
   - Task 20 (Bagel): The sentiment was marked as positive based
     on the spreads, not the bagel itself.
   - Occurrences: 14

3. **Overlooking Specific Details or Misjudging Importance**:
   - Task 12 (Edamame): The sentiment was marked as positive based
      on the aspect of being free, overlooking the neutral
     sentiment towards the edamame itself.
   - Occurrences: 1

In summary, the primary fault categories and their occurrences are
    :
- Misinterpretation of Context or Negation: 7 occurrences
- Incorrect Sentiment Attribution: 14 occurrences
- Overlooking Specific Details or Misjudging Importance: 1
    occurrence

These categories highlight challenges in accurately interpreting
```

```
    nuanced sentiment, especially when context or comparative
    language is involved.
```

Error Classes - Dynamic-Few-Shot - Unstructured

```
The false decisions can be grouped into distinct fault categories
    as follows:

1. **Misinterpretation of Contextual Sentiment**:
   - Task 1 (service): The positive sentiment was misinterpreted
     as neutral due to overlooking the overall positive context.
   - Task 3 (single drink): The neutral sentiment was
     misinterpreted as negative due to focusing on service
     issues rather than the item itself.
   - Task 4 (drink list): The neutral sentiment was misinterpreted
      as positive due to misinterpreting a suggestion for
     improvement as positive sentiment.
   - Task 5 (dinner): The negative sentiment was misinterpreted as
      neutral due to focusing on external factors rather than
     the sentiment towards the term itself.
   - Task 7 (Food): The neutral sentiment was misinterpreted as
     positive due to overemphasis on descriptive adjectives
     without negative context.
   - Task 8 (Scene): The neutral sentiment was misinterpreted as
     positive due to misinterpreting descriptive elements as
     positive sentiment.
   - Task 9 (ravioli): The positive sentiment was misinterpreted
     as negative due to focusing on negative aspects without
     considering the overall sentiment.
   - Task 10 (sauce): The positive sentiment was misinterpreted as
      neutral due to not recognizing implicit positive sentiment
     .
   - Task 11 (menu): The neutral sentiment was misinterpreted as
     negative due to associating environmental factors with the
     term itself.
   - Task 12 (burgers): The neutral sentiment was misinterpreted
     as positive due to misinterpreting admiration as positive
     sentiment.
   - Task 13 (appropriately earthy mushroom polenta): The neutral
     sentiment was misinterpreted as positive due to
     misinterpreting descriptive accuracy as positive sentiment.
   - Task 14 (meal): The negative sentiment was misinterpreted as
     neutral due to focusing on service issues rather than the
     item itself.
   - Task 15 (maitre): The neutral sentiment was misinterpreted as
      negative due to misinterpreting a factual statement as
     negative sentiment.
```

```
   - Task 16 (brunch): The neutral sentiment was misinterpreted as
      negative due to associating service issues with the term
     itself.
   - Task 17 (pizza): The positive sentiment was misinterpreted as
      negative due to misinterpreting a desire for variety as
     negative sentiment.
   - Task 18 (dinner): The neutral sentiment was misinterpreted as
      positive due to misinterpreting expectations as positive
     sentiment.
   - Task 19 (banquet menu): The neutral sentiment was
     misinterpreted as negative due to associating
     disappointment with the term itself.
   - Task 20 (waiters): The positive sentiment was misinterpreted
     as negative due to focusing on service capacity issues.

Occurrences: 19

2. **Incorrect Sentiment Polarity**:
   - Task 2 (fries): The positive sentiment was incorrectly
     identified as negative due to misinterpreting comparative
     language.

Occurrences: 1

In summary, the majority of the incorrect decisions fall under the
     category of "Misinterpretation of Contextual Sentiment" with
    19 occurrences, where the LLM failed to accurately interpret
    the overall sentiment context or misattributed sentiment based
     on surrounding text. There was only 1 occurrence of "
    Incorrect Sentiment Polarity," where the sentiment was
    directly opposite to the actual sentiment.
```

Error Classes - Dynamic-Few-Shot - Structured

```
Based on the provided examples, we can categorize the incorrect
    sentiment analysis decisions into distinct fault categories.
    Here are the identified categories along with the frequency of
     their occurrence:

1. **Misinterpretation of Context or Modifier**:
   - Task 1 (service: neutral instead of positive)
   - Task 3 (single drink: negative instead of neutral)
   - Task 4 (drink list: positive instead of neutral)
   - Task 5 (dinner: neutral instead of negative)
   - Task 6 (grill: positive instead of neutral)
   - Task 7 (Food: positive instead of neutral)
   - Task 8 (Scene: positive instead of neutral)
   - Task 9 (ravioli: negative instead of positive)
```

```
   - Task 10 (sauce: neutral instead of positive)
   - Task 11 (menu: negative instead of neutral)
   - Task 12 (burgers: positive instead of neutral)
   - Task 13 (appropriately earthy mushroom polenta: positive
      instead of neutral)
   - Task 14 (meal: neutral instead of negative)
   - Task 15 (maitre: negative instead of neutral)
   - Task 16 (brunch: negative instead of neutral)
   - Task 17 (pizza: negative instead of positive)
   - Task 18 (dinner: positive instead of neutral)
   - Task 19 (banquet menu: negative instead of neutral)
   - Task 20 (waiters: negative instead of positive)

   **Frequency**: 20 occurrences

2. **Incorrect Sentiment Polarity**:
   - Task 2 (fries: negative instead of positive)

   **Frequency**: 1 occurrence

These categories encompass the primary types of errors made in the
     sentiment analysis tasks. The most common fault category is
    the misinterpretation of context or modifier, where the
    sentiment towards a term is incorrectly identified due to
    misunderstanding the context or the impact of modifiers in the
     sentence. This category includes errors where a positive or
    negative sentiment is assigned when it should be neutral, or
    vice versa, and also includes cases where the sentiment is
    completely opposite to what the context suggests. The
    incorrect sentiment polarity category is less frequent and
    involves cases where the sentiment is identified correctly as
    either positive or negative but is assigned the wrong polarity
    .
```

Error Classes - Logistic Regression

```
Based on the provided false decisions, we can categorize the
    errors made by the Logistic Regression Machine Learning Model
    into distinct fault categories. Here are the identified
    categories along with the frequency of each:

1. **Misinterpretation of Context or Negation**:
   - Task 1: The positive aspect of 'tables' is misinterpreted as
      negative due to possibly the model focusing on "wish they
      had more" without understanding the context.
   - Task 5: The model fails to recognize the implied negative
      sentiment towards 'meal' due to the lack of direct negative
       words, even though the context suggests dissatisfaction.
```

```
   - Task 7: The positive experience with the 'manager' is
     incorrectly marked as negative, possibly due to the model
     not capturing the overall positive context.
   - Task 17: The 'waiter' being promptly available at the start
     but then disappearing is a nuanced context that the model
     misinterprets as entirely negative.
   - Task 19: The model incorrectly assesses 'meat' as negative,
     failing to understand the context where the meat, onions,
     and cheese were good.
   - Occurrences: 5

2. **Incorrect Sentiment Polarity**:
   - Task 9: The model fails to recognize the negative sentiment
     towards 'eggs' being cold, marking it as neutral.
   - Task 11: The model marks 'salmon' as neutral despite the
     negative context of not being able to afford it.
   - Task 13: The model incorrectly marks 'portions' as negative
     despite the context indicating that large portions are a
     positive aspect.
   - Task 20: 'Rice pudding' is marked as negative despite no
     direct negative sentiment expressed towards it.
   - Occurrences: 4

3. **Failure to Recognize Positive Aspects**:
   - Task 3: The model marks 'menu' as neutral despite the
     positive experience described.
   - Task 6: 'Bagels' are marked as neutral despite the
     description of their perfect softness and crispiness, which
      is positive.
   - Task 10: The model fails to recognize the positive sentiment
     towards 'entrees' provided by the diverse array mentioned.
   - Task 12: 'Appetizers' are marked as neutral despite being
     described positively.
   - Task 14: The model marks 'chole' as neutral despite being
     part of a list of liked items.
   - Task 15: 'Delivery' is marked as neutral despite the context
     suggesting a positive aspect due to high demand.
   - Task 18: 'Red pepper hummus' is marked as neutral despite
     being offered as a compensatory gesture, which is positive.
   - Occurrences: 7

4. **Failure to Recognize Neutral Aspects**:
   - Task 16: The model incorrectly assigns a positive sentiment
     to 'texture', which is more of a neutral description.
   - Occurrences: 1

5. **Incorrect Neutral Sentiment Assignment**:
   - Task 2: The model marks 'food' as negative despite the
     context suggesting a neutral stance due to untouched food
```

```
    and drinks.
  - Task 4: 'Goat cheese' is marked as positive despite being
    part of a neutral description of the dish components.
  - Task 8: The model marks 'Scene' as neutral despite the
    context suggesting a positive aspect of popularity among
    locals.
  - Occurrences: 3

In summary, the errors can be grouped into five main categories,
    with the most common error being the failure to recognize
    positive aspects, occurring 7 times. Misinterpretation of
    context or negation and incorrect sentiment polarity are also
    significant, with 5 and 4 occurrences respectively. Incorrect
    neutral sentiment assignment and failure to recognize neutral
    aspects are less common, with 3 and 1 occurrences respectively
    .
```

A.3 Price Prediction

Error Analysis

Structured Explanation Overview - Zero-Shot

attribute	count	mean	std.
km_driven	400	0.254	0.059
year	400	0.251	0.047
fuel	399	0.157	0.022
transmission	363	0.139	0.029
engine	296	0.115	0.033
mileage	101	0.102	0.020
owner	182	0.100	0.019
max_power	200	0.088	0.039
seller_type	8	0.075	0.027
seats	56	0.074	0.027
torque	4	0.063	0.025
name	0	0.000	0.000

Table A.4: Structured Explanation Overview - Zero-Shot - Price Prediction

Structured Explanation Overview - Fix-Few-Shot

Attribute	Count	Mean	Std.
km_driven	400	0.228	0.049
year	400	0.167	0.036
fuel	389	0.145	0.047
transmission	274	0.129	0.035
engine	390	0.127	0.029
owner	128	0.126	0.044
mileage	389	0.125	0.033
max_power	321	0.089	0.038
seller_type	33	0.083	0.030
torque	47	0.069	0.030
seats	20	0.060	0.026
name	0	0.000	0.000

Table A.5: Structured Explanation Overview - Fix-Few-Shot - Price Prediction

Structured Explanation Overview - Dynamic-Few-Shot

Attribute	Count	Mean	Std.
km_Driven	400	0.250	0.057
year	404	0.184	0.058
owner	277	0.142	0.054
engine	322	0.124	0.033
fuel	202	0.122	0.026
transmission	218	0.114	0.033
max_Power	329	0.108	0.044
mileage	320	0.102	0.031
seller_Type	59	0.100	0.035
name	5	0.100	0.000
torque	139	0.094	0.041
seats	131	0.073	0.033

Table A.6: Structured Explanation Overview - Dynamic-Few-Shot - Price Prediction

Price Error Classes - Lasso Regression

```
Analyzing the false predictions provided, we can categorize the
    errors into distinct fault categories based on the
    characteristics of the predictions and the actual prices. Here
     are the identified categories:

1. **High-Value Bias**: This category includes predictions where
    the model significantly overestimates the value of high-end or
     luxury vehicles. This could be due to the model placing too
    much weight on features like brand, engine size, or power,
    without adequately considering depreciation factors or market
    demand for these vehicles.

    - Occurrences: Task 8 (BMW X1 sDrive20d M Sport), Task 18 (
       Jeep Compass 1.4 Sport), Task 20 (Audi A3 40 TFSI Premium)
    - Total: 3

2. **Underestimation of Popular Models**: Here, the model
    underestimates the value of cars that are popular in the
    market, possibly due to their reliability, brand loyalty, or
    resale value. This could be because the model does not
    adequately account for the intangible aspects that contribute
    to a car's market value.

    - Occurrences: Task 11 (Maruti Swift VVT ZXI), Task 19 (Tata
       Tiago 1.2 Revotron XT)
    - Total: 2

3. **Overestimation of Mileage and Engine Power**: This category
    includes errors where the model overestimates the price based
    on high mileage or engine power, without considering other
    depreciating factors such as the car's age, km driven, or
    overall condition.

    - Occurrences: Task 1 (Ford Endeavour 3.0L 4X2 AT), Task 7 (
       Hyundai Xcent 1.2 CRDi E)
    - Total: 2

4. **Misjudgment of Market Trends**: These are errors where the
    model fails to accurately predict prices due to not
    considering current market trends, such as the increasing
    demand for SUVs and crossovers or the depreciation rate of
    certain models.

    - Occurrences: Task 9 (Mahindra Bolero 2011-2019 SLE), Task 10
        (Ford Figo Aspire 1.5 TDCi Titanium), Task 13 (Maruti
       Ertiga ZDI), Task 14 (Renault Duster 110PS Diesel RxZ),
       Task 15 (Tata New Safari DICOR 2.2 VX 4x2)
    - Total: 5
```

5. **Overestimation for Older Models**: This includes predictions where the model overestimates the value of older vehicles, possibly due to not adequately penalizing the age or km driven .

   - Occurrences: Task 4 (Ford Figo Diesel EXI Option), Task 5 (Toyota Corolla DX), Task 6 (Chevrolet Spark 1.0 LT Option Pack w/ Airbag), Task 12 (Maruti Alto 800 LXI), Task 16 (Toyota Innova 2.5 G (Diesel) 7 Seater BS IV), Task 17 (Tata Sumo MKII Turbo 2.0 LX)
   - Total: 6

6. **General Overestimation/Underestimation**: This category is for predictions that don't necessarily fit into the more specific categories above but where the model still made significant errors in either direction.

   - Occurrences: Task 2 (Maruti Wagon R LXI Minor), Task 3 (Maruti Zen Estilo LXI BS IV)
   - Total: 2

Price Error Classes - Zero Shot – Structured

Analyzing the provided data, the incorrect predictions can be grouped into distinct fault categories based on the nature of the errors observed. Here are the identified categories along with the frequency of their occurrence:

1. **Inadequate Consideration of Vehicle Age** (5 occurrences): This category includes predictions where the model year of the vehicle significantly impacts its value, but the prediction does not adequately account for the depreciation or value retention associated with the vehicle's age. Examples include Task 4 (Maruti Zen Classic, 1998), Task 8 (Mahindra Bolero SLX , 2007), and Task 11 (Maruti Zen Estilo 1.1 LX BSIII, 2007).

2. **Overemphasis on Mileage and Engine Specifications** (3 occurrences): Predictions that overly focus on the vehicle's mileage or engine specifications, leading to inaccurate pricing. For instance, Task 5 (Toyota Corolla DX, 2005) and Task 18 (Chevrolet Spark 1.0 LT Option Pack w/ Airbag, 2009) fall into this category.

3. **Neglect of Market Trends and Brand Value** (4 occurrences): These predictions fail to consider the brand's market value or specific trends affecting certain models' resale value. Task 6 (Mercedes-Benz New C-Class C 200 Kompressor Elegance AT, 2007) and Task 17 (Fiat Grande Punto 1.3 Emotion, 2011) are

```
examples where the brand's value or model-specific trends
    might have been overlooked.

4. **Misjudgment of Vehicle Condition Based on Ownership and Usage
    ** (2 occurrences): Incorrect predictions due to not properly
    evaluating the vehicle's condition inferred from the number of
     previous owners or the kilometers driven. Task 7 (Hyundai
    Verna 1.6 SX, 2011) and Task 15 (Ford Figo Diesel ZXI, 2014)
    are examples where the condition inferred from ownership and
    usage was likely misjudged.

5. **Inaccurate Weighting of Attributes** (4 occurrences): These
    are cases where the prediction model incorrectly weighted
    certain vehicle attributes, leading to a significant deviation
     from the actual price. For example, Task 9 (Mahindra Bolero
    2011-2019 SLE) and Task 14 (Maruti SX4 ZXI AT, 2011) show
    signs of attributes being weighted in a way that doesn't align
     well with their actual impact on the vehicle's market price.

6. **Lack of Consideration for Missing Information** (2
    occurrences): Predictions that did not adequately account for
    missing information, which could have led to overestimation or
     underestimation of the vehicle's value. Task 2 (Maruti Swift
    VDI BSIV W ABS, 2011) and Task 12 (Hyundai Santro Xing XL,
    2006) are instances where missing data (e.g., mileage, engine
    specs) might have been critical for an accurate prediction.

In summary, the fault categories and their occurrences are as
    follows:
- Inadequate Consideration of Vehicle Age: 5
- Overemphasis on Mileage and Engine Specifications: 3
- Neglect of Market Trends and Brand Value: 4
- Misjudgment of Vehicle Condition Based on Ownership and Usage: 2
- Inaccurate Weighting of Attributes: 4
- Lack of Consideration for Missing Information: 2
```

Price Error Classes - Zero-Shot – Unstructured

```
Analyzing the false predictions provided, we can categorize the
    errors into distinct fault categories based on the reasoning
    provided for each prediction and the deviation from the actual
     price. Here are the identified categories and the frequency
    of their occurrence:

1. **Misjudgment of Vehicle Attributes' Impact**:
   - This category includes predictions where the LLM incorrectly
      assessed the impact of certain vehicle attributes (e.g.,
```

year, mileage, engine size, power output) on the selling price.
- Occurrences: Task 1, Task 3, Task 5, Task 6, Task 7, Task 8, Task 9, Task 10, Task 11, Task 12, Task 13, Task 14, Task 15, Task 16, Task 17, Task 18, Task 19, Task 20

2. **Overreliance on Incomplete Data**:
- Predictions in this category were made with significant missing information (e.g., missing engine size, power output, mileage), leading to inaccurate price estimations.
- Occurrences: Task 2, Task 12

3. **Overestimation of Model Popularity or Brand Value**:
- This involves predictions where the LLM overestimated the impact of a car's brand or model popularity on its selling price.
- Occurrences: Task 16

4. **Underestimation of Market Demand or Trends**:
- In these cases, the LLM failed to accurately account for current market demand or trends that could affect the selling price of the vehicle.
- No clear occurrences identified explicitly, but it could be an underlying factor in several misjudgments.

5. **Incorrect Assessment of Ownership and Seller Type Impact**:
- Predictions where the LLM did not correctly evaluate how the type of seller (individual vs. dealer) or the number of previous owners would influence the selling price.
- Occurrences: Task 4, Task 7, Task 20

6. **Neglect of Vehicle Condition or Features**:
- This category includes predictions where the LLM did not adequately consider the condition of the vehicle or specific features that could significantly impact its value .
- No explicit occurrences identified, but it could be a contributing factor in the misjudgment of vehicle attributes' impact.

Summary of Fault Categories and Frequencies:
- Misjudgment of Vehicle Attributes' Impact: 18 occurrences
- Overreliance on Incomplete Data: 2 occurrences
- Overestimation of Model Popularity or Brand Value: 1 occurrence
- Incorrect Assessment of Ownership and Seller Type Impact: 3 occurrences

Note: The categorization is based on the explanations provided for each prediction. Some predictions may fit into multiple categories, but for clarity, they have been placed in the most dominant category based on the provided explanation.

Price Error Classes - Fix-Few-Shot - Structured

```
Based on the provided examples of wrong predictions, we can
    categorize the faults into distinct categories. These
    categories are derived from the common patterns observed in
    the prediction errors:

1. **High Mileage Overvaluation**: Predictions that significantly
    overestimate the value of vehicles with high mileage, possibly
     due to an overemphasis on other factors like engine size,
    power, or brand, without adequately accounting for the
    depreciation effect of high mileage.
    - Occurrences: Task 3, Task 6, Task 7, Task 8, Task 9, Task
        10, Task 11, Task 12, Task 13, Task 14, Task 15, Task 16,
        Task 17, Task 18, Task 19, Task 20

2. **Underestimation of Recent Model Years**: Predictions that
    undervalue vehicles from more recent model years, potentially
    due to not placing enough importance on the year attribute
    relative to other factors.
    - Occurrences: Task 11, Task 12, Task 13, Task 14, Task 15,
        Task 20

3. **Overlooking Vehicle Condition and Ownership History**: Faulty
     predictions that do not seem to adequately consider the
    vehicle's condition or the number of previous owners, which
    can significantly impact a vehicle's market value.
    - Occurrences: Task 7, Task 15, Task 16

4. **Misjudgment of Vehicle Type and Brand Value**: Incorrect
    predictions that fail to recognize the inherent value
    associated with certain vehicle types (e.g., SUVs, luxury
    brands) or specific brands, leading to significant prediction
    errors.
    - Occurrences: Task 18, Task 19, Task 20

5. **Inaccurate Valuation of Unique or Less Common Features**:
    Errors arising from incorrect assessments of vehicles with
    unique or less common features (e.g., engine size, power,
    special editions), possibly due to a lack of comparable data
    points or an underestimation of their value contribution.
    - Occurrences: Task 18, Task 19, Task 20

6. **Overestimation of Older Vehicles**: Predictions that
    overvalue older vehicles, possibly due to an overemphasis on
```

```
    attributes like engine size, power, or maintenance (if assumed
     well-maintained) without considering the significant
    depreciation effect over time.
     - Occurrences: Task 7, Task 8, Task 10

7. **Neglecting Market Trends and Demand**: Errors that may result
     from not considering current market trends and consumer
    demand, which can greatly affect the resale value of certain
    models or types of vehicles.
     - Occurrences: Task 11, Task 12, Task 13, Task 14, Task 15,
        Task 20

8. **Incomplete Data Handling**: Faulty predictions in scenarios
    where key information is missing (e.g., mileage, engine size,
    power), indicating a potential over-reliance on available data
     without adequately accounting for the impact of missing
    attributes.
     - Occurrences: Task 7, Task 16

These categories highlight the multifaceted nature of vehicle
    price prediction errors, suggesting areas for model
    improvement such as better accounting for depreciation factors
    , more accurately valuing recent models and unique features,
    and improving the handling of incomplete data.
```

Price Error Classes - Fix-Few-Shot - Unstructured

```
Based on the provided examples of wrong predictions, we can
    categorize the faults into distinct categories as follows:

1. **Overestimation of Market Value**:
   - Task 1: Overestimated the impact of high kilometers and
      diesel fuel type.
   - Task 3: Overestimated the value based on year and mileage.
   - Task 5: Overestimated based on engine size and power.
   - Task 7: Overestimated due to lack of information on crucial
      parameters like engine size, power, and mileage.
   - Task 9: Overestimated based on year and engine specifications
      .
   - Task 11: Overestimated the impact of the car's features and
      condition.
   - Task 17: Overestimated based on year, kilometers driven, and
      fuel type.
   - Task 18: Overestimated based on engine capacity and power.
   - Task 19: Overestimated based on year, mileage, engine
      capacity, and power.
   - Task 20: Overestimated based on engine size, power, and
      seating capacity.
```

```
   - Task 24: Slightly overestimated based on low mileage and
     recent year.
   - Task 25: Overestimated the luxury brand's value and car's low
      mileage.

Occurrences: 12

2. **Underestimation of Market Value**:
   - Task 6: Underestimated the car's value despite its newer year
      and low mileage.
   - Task 10: Underestimated the value of a well-maintained older
     car with low mileage.
   - Task 12: Underestimated the value of a newer model with
     reasonable mileage.
   - Task 13: Significantly underestimated the car's market value
     based on its attributes.
   - Task 14: Underestimated the value of a relatively new car
     with low mileage.
   - Task 15: Severely underestimated the value, possibly due to
     high mileage and multiple owners.
   - Task 16: Underestimated the car's value, not accounting
     enough for its low mileage and being a diesel variant.
   - Task 23: Slightly underestimated the car's value, possibly
     not fully accounting for its low mileage and recent year.

Occurrences: 8

3. **Misjudgment of Vehicle's Desirability or Market Demand**:
   - Task 2: Possibly underestimated the desirability of the model
      despite its age.
   - Task 4: Misjudged the market demand for the model, leading to
      underestimation.
   - Task 8: Misjudged the classic car's market value, leading to
     overestimation.
   - Task 22: Slightly overestimated, possibly misjudging the
     specific model's demand.

Occurrences: 4

4. **Inadequate Consideration of Vehicle Condition or Missing
   Information**:
   - Task 7: Overestimated without considering missing crucial
     information like engine size, power, and mileage.
   - Task 16: Overestimated possibly without adequate
     consideration of the vehicle's condition or missing
     specifics.

Occurrences: 2
```

```
**Summary**:
- **Overestimation of Market Value**: 12 occurrences
- **Underestimation of Market Value**: 8 occurrences
- **Misjudgment of Vehicle's Desirability or Market Demand**: 4
    occurrences
- **Inadequate Consideration of Vehicle Condition or Missing
    Information**: 2 occurrences

These categories highlight the challenges in accurately predicting
     used car prices based on a set of attributes, especially when
     considering the nuanced factors like market demand, vehicle
    condition, and the impact of missing information.
```

Price Error Classes - Dynamic-Few-Shot - Structured

```
Based on the provided examples, the incorrect predictions can be
    grouped into distinct fault categories as follows:

1. **High Mileage Overlook**: This category includes cases where
    the prediction failed to adequately account for the high
    mileage of the vehicle, which typically decreases the vehicle'
    s value. This fault occurred in Task 3, where the car had
    150,000 km driven but was predicted to have a higher price
    than the actual.

2. **Owner Count Misjudgment**: Predictions that did not properly
    factor in the depreciation associated with the number of
    previous owners fall into this category. For instance, Task 6,
     where the car had a "Second Owner" status but was predicted
    to have a higher price than the actual.

3. **Year of Manufacture Misinterpretation**: This category
    includes predictions that inaccurately assessed the impact of
    the vehicle's year of manufacture on its price. An example is
    Task 17, where the vehicle's year (2011) was not given enough
    weight in the prediction, leading to a significant
    underestimation of its price.

4. **Kilometers Driven Underestimation**: Predictions that failed
    to properly account for the effect of the kilometers driven on
     the vehicle's value. For example, Task 18, where the car had
    91,000 km driven but was predicted to have a higher price than
     the actual.

5. **Vehicle Condition and Features Overlook**: This includes
    cases where the prediction did not sufficiently consider the
    vehicle's condition or specific features that could affect its
     price. Task 19 is an example, where the prediction
```

significantly underestimated the price, possibly due to not fully considering the vehicle's condition or features.

6. **Market Trends and Model Popularity Ignorance**: This category is for predictions that might not have considered current market trends or the popularity of specific models, which can greatly affect vehicle prices. While not explicitly mentioned in the tasks, it's a potential underlying issue in several incorrect predictions.

7. **Transmission Type Impact Misjudgment**: Incorrect predictions that did not accurately assess the impact of the transmission type (Manual or Automatic) on the vehicle's price. For example, Task 18, where the prediction might have underestimated the value added by an automatic transmission.

8. **Fuel Type Impact Misjudgment**: This includes predictions that failed to properly consider how the type of fuel (Petrol or Diesel) influences the vehicle's price. Task 20 could fall into this category if the diesel variant was undervalued in the prediction.

Frequency of Occurrence:
- High Mileage Overlook: 1 occurrence
- Owner Count Misjudgment: 1 occurrence
- Year of Manufacture Misinterpretation: 1 occurrence
- Kilometers Driven Underestimation: 1 occurrence
- Vehicle Condition and Features Overlook: 1 occurrence
- Market Trends and Model Popularity Ignorance: Potentially multiple, but not explicitly identified
- Transmission Type Impact Misjudgment: 1 occurrence
- Fuel Type Impact Misjudgment: Potentially 1 occurrence

It's important to note that these categories are inferred based on the provided examples and explanations. The actual reasons behind each incorrect prediction could vary and might involve multiple factors.

Error Classes - Dynamic-Few-Shot - Unstructured

Based on the provided examples of wrong predictions, we can categorize the faults into distinct categories. These categories are derived from the common reasons behind the inaccuracies in the predictions:

1. **Misjudgment of Vehicle Age Impact**: This category includes predictions where the model year of the vehicle was not accurately factored into the price estimation. For example,

```
   older vehicles were sometimes predicted to have higher prices
   than they actually did, not accounting sufficiently for
   depreciation.

2. **Overlooking High Mileage**: In several cases, the high
   mileage of a vehicle, which significantly impacts its value,
   was not adequately considered, leading to overestimations of
   the vehicle's worth.

3. **Owner History Misinterpretation**: The number of previous
   owners and its impact on the vehicle's price was another area
   of fault. Cars with multiple previous owners, which would
   typically lower the price, were not always adjusted
   accordingly.

4. **Incorrect Valuation of Vehicle Condition and Specifications
   **: This includes cases where specific attributes such as
   engine size, max power, and additional features (e.g.,
   automatic transmission) were either overvalued or undervalued
   in the context of their impact on the overall price.

5. **Market Trend Misalignment**: Some predictions did not align
   with general market trends, such as the demand for certain
   models or the impact of fuel type on a vehicle's resale value.

6. **Inaccurate Comparison with Example Outputs**: In some
   instances, the prediction seemed to overly rely on the example
    outputs without considering the unique aspects of the new
   task, leading to incorrect price estimations.

Occurrences:
- Misjudgment of Vehicle Age Impact: 4 times
- Overlooking High Mileage: 5 times
- Owner History Misinterpretation: 3 times
- Incorrect Valuation of Vehicle Condition and Specifications: 4
   times
- Market Trend Misalignment: 2 times
- Inaccurate Comparison with Example Outputs: 2 times

These categories and occurrences highlight the complexity of
   accurately predicting used car prices based on a set of
   features. Each vehicle's price is influenced by a combination
   of factors, including but not limited to age, mileage,
   condition, market demand, and historical ownership. The
   discrepancies in the predictions suggest a need for a more
   nuanced approach that can weigh these factors more dynamically
   .
```

Error Classes - Overview Tables

	Zero-Shot	**Fix-Few-Shot**	**Dyn.-Few-Shot**
Class 1	Overemphasis on Mileage and Engine Specifications (5)	High Mileage Overvaluation (16)	High Mileage Overlook (1)
Class 2	Inadequate Consideration of Vehicle Age (3)	Underestimation of Recent Model Years (6)	Year of Manufacture Misinterpretation (1)
Class 3	Misjudgment of Vehicle Condition Based on Ownership and Usage (4)	Overlooking Vehicle Condition and Ownership History (3)	Vehicle Condition and Features Overlook (1)
Class 4	Neglect of Market Trends and Brand Value (2)	Neglecting Market Trends and Demand (7)	Market Trends and Model Popularity Ignorance (-)
Class 5	Inaccurate Weighting of Attributes (4)	Inaccurate Valuation of Unique or Less Common Features (3)	Owner Count Misjudgment (1)
Class 6	Lack of Consideration for Missing Information (2)	Incomplete Data Handling (2)	Kilometers Driven Underestimation (1)
Class 7	-	Misjudgment of Vehicle Type and Brand Value (3)	Transmission Type Impact Misjudgment (1)
Class 8	-	Overestimation of Older Vehicles (3)	Fuel Type Impact Misjudgment (1)

Table A.7: Error Class Overview with # of occurrences - Structured Explanations

	Zero-Shot	Fix-Few-Shot	Dyn.-Few-Shot
Class 1	Misjudgment of Vehicle Attributes (18)	Misjudgment of Vehicle's Desirability or Market Demand (4)	Incorrect Valuation of Vehicle Condition and Specifications (4)
Class 2	Overreliance on Incomplete Data (2)	Inadequate Consideration of Vehicle Condition or Missing Information (2)	Market Trend Misalignment (2)
Class 3	Overestimation of Model Popularity or Brand Value (1)	Overestimation of Market Value (12)	Misjudgment of Vehicle Age Impact (4)
Class 4	Underestimation of Market Demand or Trends (-)	Underestimation of Market Value (8)	Overlooking High Mileage (5)
Class 5	Incorrect Assessment of Ownership and Seller Type Impact (3)	-	Owner History Misinterpretation (3)
Class 6	Neglect of Vehicle Condition or Features (-)	-	Inaccurate Comparison with Example Outputs (2)

Table A.8: Error Class Overview with # of occurrences - Unstructured Explanations - Price Prediction

	Lasso Regression
Class 1	High-Value Bias (3)
Class 2	Underestimation of Popular Models (2)
Class 3	Overestimation of Mileage and Engine Power (2)
Class 4	Misjudgment of Market Trends (5)
Class 5	Overestimation for Older Models (6)
Class 6	General Overestimation/Underestimation (2)

Table A.9: Error Class Overview with # of occurrences - Lasso Regression

Appendix B

Program Code / Resources

The source code, datasets, a documentation, and additional test results are available at: `https://github.com/timgutberlet/Error-Analysis-with-LLMs`. They also are included in the USB Flash Drive that comes with the thesis.

Appendix C

Prompts and Preprocessing

C.1 Preprocessing

C.2 Schema Matching

Single-Column Zero-Shot Prompt

```
zero_shot_single_column= ChatPromptTemplate.from_messages([
    ("user",
"""Question:
Table A:
{Table_A}

Table B:
{Table_B}

Task: Identify if Column-A-{Column_A} in Table A and Column-B-{
    Column_B} in Table B are matching columns. Return "True" if
    they are matching and "False" if they are not.
ONLY return "True" or "False". Do not return anything else."""),
])
```

Multi-Column Zero-Shot Prompt

```
zero_shot_multi_column= ChatPromptTemplate.from_messages([
    ("system", """Description: Please identify the matching
        columns between Table A and Table B.
     For each column in Table A, specify the corresponding column
         in Table B.
     If a column in A has no corresponding column in Table B, you
         can map it to 'None'.
```

```
     Represent each column mapping using a pair of column headers
         in a list, i.e., [Table A Column, Table B column or None
         ].
     Provide the mapping for each column in Table A and return all
          mappings in a list. Return the final result as JSON in
         the format {{"column_mappings": "<a list of column pairs
         >"}}."""),
    ("user", """Question:
Table A:
{Table_A}

Table B:
{Table_B}

Return the final result as JSON in the format {{"column_mappings":
     "<a list of column pairs>"}}. ONLY return the JSON and
    nothing else.
Answer:"""),
])
```

Multi-Column Few-Shot Prompt

```
fewshot_multi_table= ChatPromptTemplate.from_messages([
    ("user", """Example 1:
{example_1}

Example 2:
{example_2}
"""),
    ("system", """Description: Please identify the matching
        columns between Table A and Table B.
     For each column in Table A, specify the corresponding column
         in Table B.
     If a column in A has no corresponding column in Table B, you
         can map it to 'None'.
     Represent each column mapping using a pair of column headers
         in a list, i.e., [Table A Column, Table B column or None
         ].
     Provide the mapping for each column in Table A and return all
          mappings in a list. Return the final result as JSON in
         the format {{"column_mappings": "<a list of column pairs
         >"}}."""),
    ("user", """Question:
Table A:
{Table_A}

Table B:
{Table_B}
```

```
Return the final result as JSON in the format {{"column_mappings":
     "<a list of column pairs>"}}. ONLY return the JSON and
    nothing else.
Answer:"""),
])
```

Confidence Analysis Prompt

```
confidence = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """{ai_answer} """),
    ("human", """Provide a confidence score for your decision to {
        match} columns {columns}. 100% referring
    to full confidence. Take into account the provided tables and
        the similarities and differences of the specific columns.
        Please provide your answer exactly in the following format
         and do not return anything else.
Confidence: <value>%""")
])
```

Unstructured Explanation Prompt

```
unstructured_analysis = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """{ai_answer}"""),
    ("human", """Now explain concisely how and why you made your
        decision to {match} the pair {columns} and explicitly
        mention the values that had a high influence on your
        decision.""")
])
```

Error Class Instruction LLM

```
error_class_LLM = PromptTemplate(
    input_variables=["examples"],
    template="""In the following I will give you schema matching
        tasks together with column matching decisions. You are
        given additional explanation on the predicted matching
        decision for two specific columns and the actual answer
        regarding the match of the two columns.
```

```
The decision was made by an LLM. Can you please group the wrong
    decisions into distinct fault categories? Please also indicate
     how often each one occurs.
There are also some correct decisions in the examples. Please just
     use them as a reference and don’t categorize them.
{examples}
""")
```

Error Class Instruction ML

```
error_class_ML = PromptTemplate(
    input_variables=["examples"],
    template="""In the following I will give you a schema matching
         tasks together with a column matching decision, details
        about the decision and the if the columns actually match.
The decision was made by a Logistic Regression Machine Learning
    Model and the additional information shows the weights the
    model assigned to the different tokens.
Can you please group the wrong decisions into distinct fault
    categories? Please also indicate how often each one occurs.
There are also some correct decisions in the examples. Please just
     use them as a reference and don’t categorize them.
{examples}
""")
```

C.3 Sentiment Analysis

Single-Term Zero-Shot Prompt

```
prompt_single_term_zeroshot = PromptTemplate(
    input_variables=["input_text", "entity"],
    template="""{input_text}

What is the sentiment of the text towards ’{entity}’? Only respond
     with "positive", "negative" or "neutral" as one word.""")
```

Single-Term Few-Shot Prompt

```
prompt_single_term_fewshot = PromptTemplate(
input_variables=["example_1", "example_2", "example_3","input_text
    ", "entity"],
template=
"""Example 1:
{example_1}
```

```
Example 2:
{example_2}

Example 3
{example_3}

Task:
Input: {input_text}
Prompt: What is the sentiment in the text towards '{entity}'? Only
     respond with "positive", "negative" or "neutral" as one word
    .""")
```

Multi-Term Zero-Shot Prompt

```
prompt_multi_term_zeroshot = PromptTemplate(
    input_variables=["input_text",  "terms"],
    template="""Task: Analyze the sentiment of specific terms
        mentioned in a sentence.
You are required to evaluate whether the sentiment towards each
    term is 'positive', 'negative', or 'neutral'.

Sentence: "{input_text}"

Terms:
{terms}

Return the final result as JSON in the format {{"term_sentiments":
     "<a list of [term, sentiment] pairs>"}}.
ONLY return the JSON.
Answer:""")
```

Multi-Term Few-Shot Prompt

```
prompt_multi_term_fewshot = PromptTemplate(
    input_variables=["example_1", "example_2", "example_3", "
        input_text",  "terms"],
    template="""Example 1:
{example_1}

Example 2:
{example_2}

Example 3
{example_3}
```

```
Task:
Input: {input_text}
Terms:
{terms}
Prompt: Analyze the sentiment of specific terms mentioned in a
    sentence.
You are required to evaluate whether the sentiment towards each
    term is 'positive', 'negative', or 'neutral'.
Return the final result as JSON in the format {{"term_sentiments":
     "<a list of [term, sentiment] pairs>"}}.
ONLY return the JSON.""")
```

Confidence Analysis Prompt

```
confidence = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """{ai_answer}"""),
    ("human", """Now provide a confidence score for your decision.
         100% referring
    to full confidence. Take into account all relevant apsects of
        your decision. Please provide your answer exactly in the
        following format and do not return anything else.
Confidence: <value>%""")
])
```

Structured Explanation Prompt

```
structured_analysis = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """{ai_answer}"""),
    ("human", """Explain your prediction in a
structured format, listing words or word groups you used for your
    decision and how important you deemed them for your decision.
    Each word or word group should be accompanied by a a score
    between 0 and
1 that shows the importance of the attribute for the decision. All
     scores in total must add up to 1. Only include words that had
     an impact on the term's sentiment.
Return the final result as JSON like in this example:
[{{"adjective":"great","importance":"0.50"}},
{{"adjective":"liked","importance":"0.45"}},
{{"adjective":"did not satisfy","importance":"0.05"}}]
ONLY return the JSON.""")
])
```

Unstructured Explanation Prompt

```
unstructured_analysis = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """{ai_answer}"""),
    ("human", """Now explain concisely how you made your
        prediction and explicitly mention the words or word groups
         that had a high influence on your decision.""")
])
```

Error Class Instruction LLM

```
error_class_LLM = PromptTemplate(
    input_variables=["examples"],
    template="""In the following I will give you a few Aspect
        Based Sentiment Analysis tasks together with a sentiment
        decision, details about the decision and the actual
        sentiment towards each term.
The Prediction was made by an LLM.
Can you please group the wrong decisions into distinct fault
    categories? Please also indicate how often each one occurs.
There are also some correct decisions in the examples. Please just
     use them as a reference and don't categorize them.
{examples}
""")
```

Error Class Instruction ML

```
error_class_ML = PromptTemplate(
    input_variables=["examples"],
    template="""In the following I will give you a few Aspect
        Based Sentiment Analysis tasks together with the sentiment
         decision towards a specific term, details about the
        decision and the actualsentiment towards each term.
The decision was made by a Logistic Regression Machine Learning
    Model and the additional information shows the weights the
    model assigned to the different tokens.
Can you please group the wrong decisions into distinct fault
    categories? Please also indicate how often each one occurs.
There are also some correct decisions in the examples. Please just
     use them as a reference and don't categorize them.
{examples}
""")
```

C.4 Price Prediction

Zero-Shot Prompt

```
zero_shot = PromptTemplate(
input_variables=[],
template=
"""System: Based on the provided attributes of a used car listed
    below, please predict its selling price in Indian Rupees in
    the Indian market. The predicted price should be expressed
    solely as a number followed by the currency "INR".
Ensure that the output contains no additional text or characters
    beyond this specified format.
Attributes:
name: {name},
year: {year},
km_driven: {km_driven},
fuel: {fuel},
seller_type: {seller_type},
transmission: {transmission},
owner: {owner},
mileage: {mileage},
engine: {engine},
max_power: {max_power},
torque: {torque},
seats: {seats}

Required Output:
"price": <predicted price> INR

Please provide the prediction strictly adhering to the above
    instructions.""")
```

Few-Shot Prompt

```
few_shot = PromptTemplate(
input_variables=[],
template=
"""Example 1:
{example_1}

Example 2:
{example_2}

Example 3:
{example_3}
```

```
System: Based on the provided attributes of a used car listed
    below, please predict its selling price in Indian Rupees in
    the Indian market. The predicted price should be expressed
    solely as a number followed by the currency "INR".
Ensure that the output contains no additional text or characters
    beyond this specified format.
Attributes:
name: {name},
year: {year},
km_driven: {km_driven},
fuel: {fuel},
seller_type: {seller_type},
transmission: {transmission},
owner: {owner},
mileage: {mileage},
engine: {engine},
max_power: {max_power},
torque: {torque},
seats: {seats}

Required Output:
"price": <predicted price> INR

Please provide the prediction strictly adhering to the above
    instructions.""")
```

Confidence Analysis Prompt

```
confidence = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """"price": {ai_answer} INR """),
    ("human", """Provide a percentage score on how confident you
        are that your predicted price is in a range of +-20% of
        the actual price. 100% referring
    to full confidence. Take into account all diffferent aspects
        of the car that make you more or less sure that your
        prediction is in the range. Please provide your answer
        exactly like in the following format and do not return
        anything else.
Confidence: <value>%""")
])
```

Structured Explanation Prompt

```
structured_analysis = ChatPromptTemplate.from_messages([
```

```
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """{ai_answer}"""),
    ("human", """Explain your prediction in a
structured format, listing attributes of the car you used and how
    important you deemed them for your prediction. Each attribute
    should be
accompanied by the attribute value and a score between 0 and
1 that shows the importance of the attribute for the prediction.
    All scores in total must add up to 1. Only include attributes
    that had an influence on your decision.
Return the final result as JSON like in this example:
[{{"attribute":"name","importance":"0.50","value":"Mahindra XUV500
     W10 2WD"}},
{{"attribute":"km_driven","importance":"0.45","value":"70000"}},
{{"attribute":"mileage","importance":"0.05","value":"16 kmpl"}}]
ONLY return the JSON.""")
])
```

Unstructured Explanation Prompt

```
unstructured_analysis = ChatPromptTemplate.from_messages([
    ("system", """You are a helpful AI."""),
    ("human", """{user_prompt}"""),
    ("ai", """{ai_answer}"""),
    ("human", """Now explain concisely how you made your
        prediction and explicitly mention the attributes and
        values that had a high influence on your decision.""")
])
```

Error Class Instruction LLM

```
error_class_LLM_nounderover = PromptTemplate(
    input_variables=["examples"],
    template="""In the following I will give you a few price
        prediction tasks together with a prediction decision,
        details about the decision and the actual price.
A wrong prediction is a prediction that deviates by more than 20
    percent from the actual price. The Prediction was made by an
    LLM.
Can you please group the wrong decisions into distinct fault
    categories? Please also indicate how often each one occurs.
There are also some correct decisions (deviation of less than 20
    percent) in the examples. Please just use them as a reference
    and don't categorize them.
```

```
Please go deeper with your analysis and don’t use Overestimation
    or Underestimation as categories.
{examples}
""")
```

Error Class Instruction ML

```
error_class_ML = PromptTemplate(
    input_variables=["examples"],
    template="""In the following I will give you a few price
        prediction tasks together with a prediction decision,
        details about the decision and the actual price.
A wrong prediction is a prediction that deviates by more than 20
    percent from the actual price.
The decision was made by a Lasso Regression Machine Learning Model
     and the additional information shows the weights the model
    assigned to the different tokens.
Can you please group the wrong decisions into distinct fault
    categories? Please also indicate how often each one occurs.
There are also some correct decisions (deviation of less than 20
    percent) in the examples. Please just use them as a reference
    and don’t categorize them.
Please go deeper with your analysis and don’t just use
    Overestimation or Underestimation as categories.
{examples}
""")
```

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Master-/Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Er- klärung rechtliche Folgen haben wird.

Frankfurt, den 30.04.2024

Unterschrift

Tim Gutbedel