Exploring Instruction Fine-Tuned Language Models for Open Aspect-Based Sentiment Analysis: A Case Study on App Reviews

models: phi-2, llama3-8b

Ben Jonas Maria Kampmann

Supervisors:
Dr. Benedict Bender
Prof. Dr. Manfred Stede

This dissertation is submitted for the degree of
Bachelor of Science

University of Potsdam October 3, 2024

Declaration of Authorship

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are my own original work and have not been submitted in whole or in part for consideration for any other degree or qualification.

Ben Jonas Maria Kampmann
October 3, 2024

Acknowledgements

I would like to thank Dr. Bender for letting me pick this unconventional topic and giving constructive criticism. I also would like to thank Prof. Dr. Stede for taking the time to review my work as a second corrector.

Abstract

This research explores the effectiveness of fine-tuned AI language models, specifically Llama-3 and Phi-2, for Open Aspect-Based Sentiment Analysis (ABSA) on app reviews. The study addresses the increasing need for automated analysis of user-generated content due to the exponential growth in online reviews. The methodology involves fine-tuning pre-trained models using supervised learning techniques on an instruction template. These embed fetched reviews from WhatsApp and multiple other apps. We utilize an rss-feed to fetch app reviews from the apple app store and use the google Gemini model API for synthetic labeling. Evaluations are conducted on both in-domain (app reviews) and out-of-domain (restaurant reviews) data. The evaluation contains a hard evaluation with metrics like f1-score, precision, recall and a soft evaluation with the Gemini model as a judge. The research aims to assess large language models performance in identifying sentiment polarity at the aspect level, leveraging techniques like Parameter-Efficient Fine-Tuning (PEFT) to optimize resource usage during fine-tuning.

Table of contents

List of figures

List of tables

Notation

Acronyms / Abbreviations

ABSA Natural Language Processing (NLP)

ABSA Aspect-Based Sentiment Analysis: A specialized approach in sentiment analysis that focuses on identifying and evaluating the sentiments related to specific aspects or features of a product or service rather than providing an overall sentiment score.

ACSD Aspect Category Sentiment Detection: A subtask of sentiment analysis that not only identifies the sentiment expressed towards a specific aspect but also classifies that aspect into categories. This allows for a more granular analysis of sentiment across various features of a product or service.

BBH BIG-Bench Hard: A challenging subset of tasks derived from the BIG-Bench benchmark, designed to rigorously evaluate the performance of advanced language models on complex reasoning and understanding tasks.

CoT Chain of Thoughts: A reasoning process that involves breaking down complex problems into smaller, logical steps, which enhances the models ability to generate coherent and contextually accurate outputs.

Implicit Aspects Unstated or implied aspects within a text that are not explicitly mentioned but can be inferred through context during sentiment analysis. These aspects require deeper understanding and context to identify, enhancing the accuracy of sentiment assessment.

PEFT Parameter-Efficient Fine-Tuning: A method for fine-tuning large language models that optimizes the use of parameters, allowing models to adapt efficiently to new tasks with minimal resource consumption. This technique is particularly valuable for applications with limited computational resources.

SA Sentiment Analysis: The process of analyzing text to determine the emotional tone or sentiment expressed within it, categorizing it as positive, negative, or neutral.

SFT Trainer Supervised Fine-Tuning Trainer: A tool that assists in the fine-tuning of pre-trained models using labeled data, enabling models to adapt to specific tasks or domains effectively.

Pre-trained Models Transformers

BERT Bidirectional Encoder Representations from Transformers: A transformer-based model that uses bidirectional training of transformers to deeply understand the context of words in a sentence. BERT has significantly advanced the state-of-the-art in various NLP tasks, including question answering and language inference.

Gemini A dual-model artificial intelligence application, which often integrates the strengths of multiple models to improve performance and versatility in tasks such as natural language processing, image recognition, and more.

PLMs Pre-trained Language Models: Models that have been trained on large corpora of text data prior to fine-tuning on specific downstream tasks, allowing them to leverage the knowledge gained during pre-training to improve performance on various NLP applications.

RoBERTa Robustly Optimized BERT Pretraining Approach: An improvement over BERT that optimizes its pretraining procedure by using longer training times, larger mini-batches, and more data, resulting in enhanced performance on numerous NLP benchmarks.

TAS-BERT Target-Aware Sentiment BERT: A variation of BERT specifically fine-tuned for aspect-level sentiment analysis, enabling it to better identify sentiments related to specific target aspects in text.

Deep Learning Architectures

Auto-regressive Language Models Models that generate text by predicting the next word or token in a sequence based solely on the words that have been generated previously.

BiLSTM Bidirectional Long Short-Term Memory network: An extension of LSTMs that processes data in both forward and backward directions, providing a richer context for understanding sequential data.

CNN Convolutional Neural Network: A type of deep learning architecture particularly effective for image processing, but increasingly utilized in NLP for tasks such as text classification and sentence modeling through convolutional layers.

Dense A fully connected layer in neural networks where each neuron in the layer is connected to every neuron in the preceding layer.

GCN Graph Convolutional Network: A variant of GNNs that applies convolutional operations on graphs, enabling efficient learning from graph-structured data.

GNN Graph Neural Network: A type of neural network specifically designed to work directly with graph-structured data, making it adept at tasks involving relationships and interactions among entities.

LSTM Long Short-Term Memory network: An advanced type of RNN capable of learning long-term dependencies in sequential data.

RNN Recurrent Neural Network: A type of neural network designed to recognize patterns in sequences of data, such as time series or text.

Transformer A deep learning architecture that employs self-attention mechanisms to process input data.

Attention Mechanisms

Grouped-Query Attention A novel attention mechanism that optimizes performance by grouping similar queries during attention computation, allowing for more efficient processing and better utilization of model resources. This method enhances scalability in transformer models.

Query, Key, and Value Projection Layers Layers employed in transformer models that facilitate the attention mechanism. Queries are compared to keys to calculate attention scores, which are then used to weight the corresponding values, allowing the model to focus on relevant information within input sequences.

Chapter 1

Introduction

1.1 Background

In recent years, a sharp increase in user-generated content like reviews on online catering platforms or e-commerce products has caught the interest of data scientists all around the world. These data have the potential to leverage businesses decisions. Analyzed correctly it provides a great reference for consumers and businesses. For instance, in the domain of catering, a large amount of accurate and objective user reviews helps merchants improve their catering products and customers to make the right decision [64]. Similary, an analysis of reviews from e-commerce platforms like Amazon or the Apple App Store provides valuable insights for product/app owners, who may then be able to improve the product or service, and start more user-adequate marketing campaigns [63].

A recent study conducted in 2022 highlights the benefit of user evaluated feedback. The authors analyzed 2084 user reviews from 1508 users on TripAdvisor and came to the conclusion, that the gained insights can improve service quality and provide sustainable economic growth for the tourism industry. [26]

However, due to the large amount of data, it is challenging to manually extract valuable insights. That's why an urgent need to analyze data by computer developed. The deployment of complex algorithms is more efficient and thus saves on labor costs. Algorithms based on the research field of sentiment analysis and more in depth Aspect-Based Sentiment Analysis (ABSA), which was used in the mentioned study from 2022 ([26]) try to solve this problem and thereby targeting one of the essential tasks of natural language processing.

The objective of sentiment analysis is to use techniques, such as text mining to correctly identify the sentiment polarity of a given text. This task can be done at text level, sentence level and aspect level. [64] Assuming that a text or review of a product, service, or app has different sentiment polarity toward multiple aspects, a fine-grained approach known as Aspect-Based Sentiment Analysis gained increasing attention in the past decade [46].

1.2 Motivation

The motivation for conducting research on Aspect-Based Sentiment Analysis (ABSA) stems from the rising volume of reviews on apps, which presents a valuable source of insights for app-developers. ABSA allows a more nuanced understanding of sentiments expressed within user reviews by identifying specific aspects of entities and their associated sentiments, thus offering fine-grained sentiment analysis that goes beyond the general sentiment of the entire review [62, 23]. Therefore developers gain a more detailed insight and can act more specific to certain problems. This is crucial in an era where user feedback and reviews significantly influence consumer behavior and business decisions.

The need for sophisticated ABSA techniques for app developers is driven by the challenges posed by traditional sentiment analysis methods, which often fails to capture the complex, multi-faceted nature of user opinions. Existing approaches often struggle with issues like implicit sentiment[1], multi-label[2] or opeb-label classification[3], and out-of-vocabulary terms, making it difficult to accurately extract and classify sentiments related to specific aspects [48, 47, 11].

Furthermore, advancements in deep-learning and language models opened new opportunities for improving ABSA. Models such as BERT, have demonstrated remarkable capabilities in capturing context making them suitable for fine-tuning on ABSA tasks. The application of these models to ABSA can lead to significant improvements in both aspect extraction and sentiment classification, addressing previous limitations of traditional machine learning techniques [45].

[1] sentiment expressed in a text without using explicit opinion words like "good" or "bad"

[2] several labels (e.g. aspects or categories) are assigned to a single instance

[3] no defined pool of labels to choose from (e.g. implicit aspects or categories)

Thereby the motivation for this research is also anchored in the need to leverage advanced language models to overcome the existing challenges in ABSA, by such means providing more accurate and detailed sentiment analysis on all domains. The integration of LMs (language models) and the continuous evolution of transformer-based models promise to significantly enhance the effectiveness of ABSA, offering valuable insights for a wide range of applications from consumer feedback to market analysis.

1.3 Research Objective

Research comprises many novel techniques to use transformer models for Aspect-Based Sentiment Analysis. Although these models show sufficient results, they often struggle with several challenges, which are adressed in 2.2. We want to test, how supervised fine-tuned state of the art language models (llam3 and phi-2) perform on the ABSA task. Specifically we want to explore, if fine-tuning those models is a valid approach to perform ABSA on app reviews. To further explore the sustainability of our approach, we conduct evaluations of our models on multiple other domains.

Chapter 2

Background

2.1 Aspect-Based Sentiment Analysis (ABSA)

Aspect-based sentiment analysis (ABSA) belongs to fine-grained sentiment analysis (SA). Traditional sentiment analysis focuses on identifying the overall sentiment polarity (positive, negative, or neutral) of a text. However this method isn't as effective, when different sentiment polarities towards multiple aspects are presented. ABSA adresses this limitation. It delves one step deeper than SA and aims at detecting the polarity of an entity or entity's aspect within a text. For instance, an entity can be a specific app and an entity's aspect could be a specific function of it. The aspect can be mentioned explicitly within the text or implicitly implied by the context[35, 55]. A more fine-grained distinction of aspects differs between aspects and categories. Categories represent latent higher-level generalizations of the aspect. Within the app domain, a category could be "interface", while the aspect is "position of search bar" [9] [62, 23].

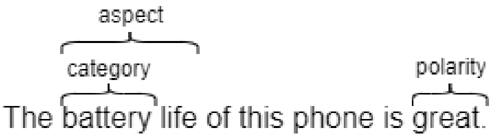


Fig. 2.1 ABSA (ACSD) example.

ABSA involves the detection of sentiments towards aspects. Traditional ABSA tasks focus on a subset of the following 4 components [62]:

- **Aspect Term (a)**: A specific entity mentioned explicit or implicit in the text.
- **Aspect Category (c)**: A broader term, that the aspect term falls under.

- **Opinion Term (o)**: The term that expresses the sentiment of the aspect, like "magnificant".
- **Sentiment Polarity (p)**: An classification of the sentiment towards an aspect, usually "positive", "neutral" or "negative"

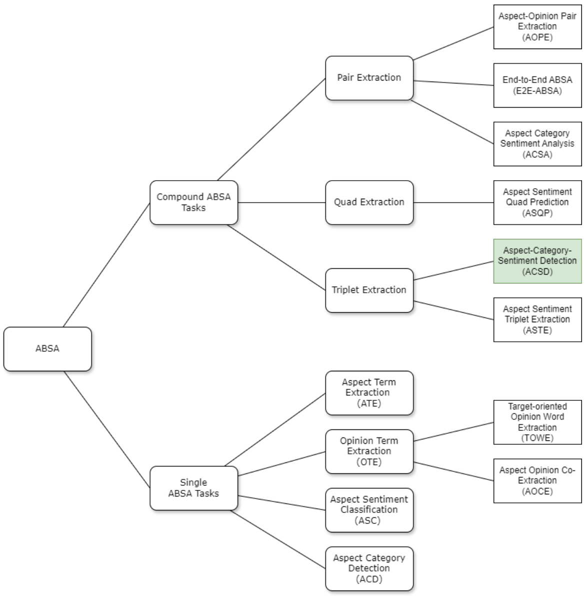

ABSA
Compound ABSA Tasks
Single ABSA Tasks
Pair Extraction
Quad Extraction
Triplet Extraction
Aspect-Opinion Pair Extraction (AOPE)
End-to-End ABSA (E2E-ABSA)
Aspect Category Sentiment Analysis (ACSA)
Aspect Sentiment Quad Prediction (ASQP)
Aspect-Category-Sentiment Detection (ACSD)
Aspect Sentiment Triplet Extraction (ASTE)
Aspect Term Extraction (ATE)
Opinion Term Extraction (OTE)
Aspect Sentiment Classification (ASC)
Aspect Category Detection (ACD)
Target-oriented Opinion Word Extraction (TOWE)
Aspect Opinion Co-Extraction (AOCE)

Fig. 2.2 ABSA taxonomy

With a sharp increase in user generated data, the need of an effective method to analyze user data, such as customer reviews rises among corporations. ABSA provides valuable insights for these, creating the foundation for a detailed analysis of big data:

- **Granular insights**: By focusing on specific aspects, ABSA enables a more detailed analysis of the data, giving companies useful insights on the factors that influence human behavior
- **Enhanced Decision-Making**: As a result of granular insights, companies can make more informed decisions by reacting to aspect specific insights.
- **Improved User Experience**: Enhanced decision-making capabilities enable organizations to align their offerings more closely with customer expectations, thereby improving the overall user experience.

With advancements in deep learning and natural language processing (NLP) the field of ABSA reached new hights. Pre-trained language models (PLMs) like BERT and

RoBERTa facilitated the creation of new ABSA models, with higher accuracy. These models open the field of Aspect-based sentiment analysis for more complex tasks including aspect category sentiment triplet detection (ACSD) [62].

We conduct our research on the compound ABSA task ACSD 2.2 with deep neural networks. Some studies [52, 60] refer to Aspect Category Sentiment Triplet Extraction, which is essentially the same task. ACSD aims to detect the aspect, its broader category and its sentiment polarity (positive, neutral, negative). Wan et al (2020) [52] proposed this tasks and thereby leveraged the TAS-BERT method to capture dependencies between categories, aspects and sentiments. Their approach demonstrated remarkable success in the SemEval-2015 [41] and SemEval-2016 [40] restaurant dataset.

2.2 Challenges of ABSA

The core challenges in Aspect-based Sentiment Analysis involve identifying pertinent aspects, establishing relational mappings between these aspects, understanding their interactions and dependencies, and considering contextual-semantic relationships. These elements are crucial for enhancing sentiment accuracy and predicting the dynamic evolution of sentiment[38].

As more and more ABSA models rely on deep neural networks, which require training data, finding a suitable dataset keeps a core challenge to researchers [23]. Evaluating the dataset quality is about assessing the domain diversity and data difficulty for ABSA. Currently there is no scientific way to asses both. [13]
Another challenge, influenced by poor training data of ABSA models relying on Machine Learning, is the correct identification of implicit aspects [19]. Furthermore, limited training data and an ambiguous connection between syntactic and semantic features are challenges for many ABSA models [20].
Researchers have also tackle the problem of an accurate identification of an aspects polarity in cases, where multiple sentiments towards an aspect are given and the provided sentiments are interdependent [62].

Many algorithms are constrained by a strict corpus of labels and thus domain dependent. We call such models "non-open". A lot of these perform great with high accuracy, but are limited in there use, due their domain-dependence. An implicit and open labeling without a strict corpus of labels, would allow a domain unspecific utilization of ABSA models. Implicit labels can be aspects or sentiments expressed in a text without using explicit words. For example, explicit sentiments are words like "good" or "bad", while implicit sentiments can only be derived with reasoning from the context. A model, that isn't constrained by a pool of labels, a number of labels per instance and a set of implicit or explicit labels is thereby called "open ABSA model". Although we acknowledge, that the sentiment labeling of instances with "positive", "neutral" and "negative" isn't "open" in our terms, we still call a such a model "open", because the sentiment labeling is domain-independent. A method, that works equally well on different domains or languages is still pending [4, 62, 24, 48, 47, 11].

However, once all challenges have been solved, the successful integration of ABSA models keeps a challenge for researchers [62].

2.3 Transformer based pretrained Language Models

Today's well known language models like ChatGPT and Gemini are based on a transformer architecture. The architecture was introduced by the famous paper "Attention is all you need" [51]. They represent a powerful tool to map semantic relationships between words. Although model architectures vary, most transformers consist of two main parts. The first is a Encoder block, which creates a semantic representation of the training vocabulary. The second block comprises a decoder, which allows to generate new language sequences. A transformer processes text in 4 main steps.

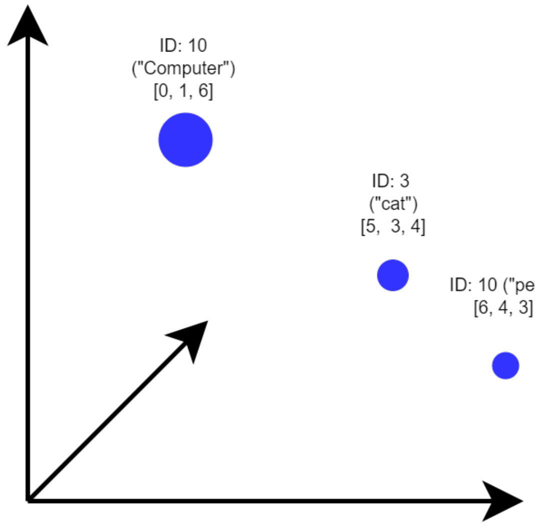


Fig. 2.3 Vector embeddings example in 3-dim space

The first step, includes the transformation of text into high dimensional vectors. This incorporates the tokenization. It assigns an ID to every word or sub-word. To further capture the semantics of these tokenized words, tokens are transformed into contextual vectors, known as embeddings. Each vector attribute represents a semantic attribute. Therefore words that are more closely related like "cat" and "pet" will be more closely located in the vector space. Which words are closely related is determined in the pre-training phase of the model, where the model tries to predict related words within a text.

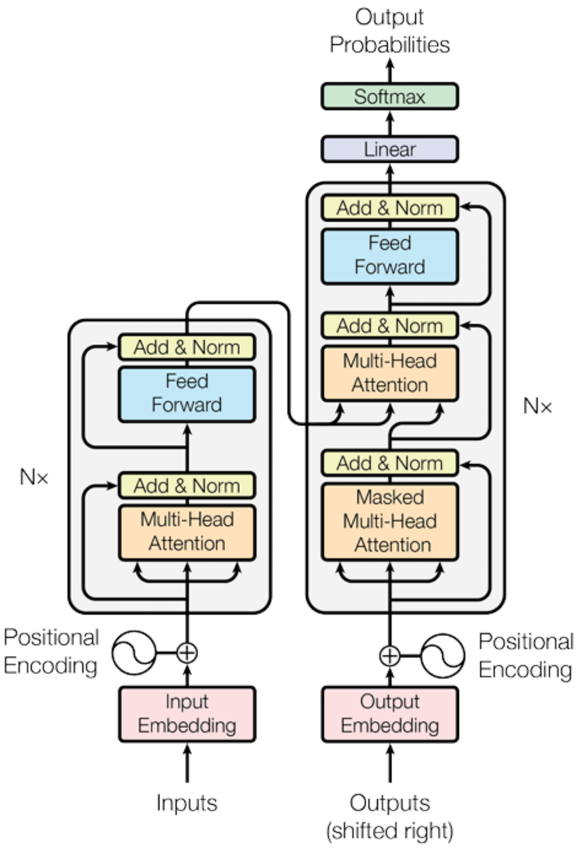


Fig. 2.4 transformer architecture
source: attention is all you need paper [51]

The tranformer layer processes the entire input sequence in parallel. It comprises the self-attention mechanism and feedforward neural network.

The attention layers within the encoder block try to examine, how a token's meaning is influenced by other given tokens. This results in a better relational representation of the input tokens within the embedding space.

In the decoder block, the attention layer allows next word prediction. Therefore, positional encoding layer adds another value to the embedding, marking the position within the sequence.

The Query, Key, and Value vectors are derived from the input text. The attention mechanism computes an attention score that measures the relevance of each word (key) in the input text to the word being processed (query). This score is then used to weight the importance of each word (value) in generating the output. The attention

score is calculated using a dot-product operation followed by a softmax function, which normalizes the scores across all words.
The self-attention mechanism enables each position in the input sequence to attend to all other positions, allowing the model to capture long-range dependencies and the context around each word. For instance, in the sentence "The cat sat on the mat," self-attention helps the model recognize that "sat" relates to "cat" more than to "mat."
Instead of computing a single set of attention scores, the transformer uses multiple attention heads to capture different types of relationships between words. Each head processes the input in a different way, focusing on various aspects of the word relationships. The outputs of these heads are then concatenated and linearly transformed to produce the final attention output. This multi-head attention mechanism allows the model to learn a richer set of features and better understand complex linguistic structures.
Feedforward Neural Networks do non-linear transformations. They incorporate the context aware representations of the input from the self-attention mechanism. This additional layer of complexity allows the model to learn intricate patterns in the input string. It is able to emphasize or suppress the outcomes of the attention layers. This provides better quality outputs.

2.4 Fine-Tuning of LMs

Fine-tuning a LM has the objective to enhance the capabilities of a model for a specific task, while keeping the cost low. Therefore pre-trained LMs are trained further on a small dataset of task-specific data after the initial pretraining on a broad dataset. This procedure adjust the pre-trained models weights to better predict domain specific inputs. The parameters for learning are thereby set in a way the model adapts more to the task-specific data, than to the data for pre-training. A model doesn't have to be retrained from scratch, thus saving resources. Also, this allows the model to not only consider the task specific data, when generating a response, but the pre-trained knowledge as well. Models can be fine-tuned for tasks like sentiment analysis, named entity recognition or machine translation.

A good example for fine-tuning a model is the adaption of language model for doctors assistance in generating patients reports from textual notes. While the standard model does not know certain used medical terms and can't understand healthcare specific jargon, the fine-tuned model is able to respond to the doctors notes with patients reports, showing a good way to adjust a base model to certain scenarios.

Supervised fine-tuning is a standard paradigm to fine-tuning a pre-trained language model using labeled data to do a specific task.
Once the dataset for fine-tuning is created, it is split into a training, evaluation and test set. A model doesn't understand text data and uses tokenized data. Tokenization is the break down of text into smaller subword units (tokens). This tokens are a numeral representation of the actual subwords. In the fine-tuning process the so called Trainer uses the text data as input and labels to make predictions, calculate errors and update the weights typically via an optimization algorithm like gradient descent. The gradients represent a magnitude, which indicates how much each weight contributes to the error. The error is the difference between the models prediction of the next word and the actual next word according the the training data.

Full fine-tuning is the process of updating all models parameters. It is the same method used for pre-training. Therefore it requires a high amount of computational resources. A fine-tuning of the Falcon model with 180 billion parameters requires more than 5120GB of memory. That is the amount 8x 8x A100 80GB GPUs have combined [54].

To limit resource requirements of full fine-tuning, **Parameter-efficient fine-tuning techniques (PEFT)** have gained significant attention. This technique involves fine-tuning only a portion of the models parameters or a subset of additional parameters. This results in less parameters to store, making fine-tuning less resource hungry. A fine-tuning of the Falcon model (40 billion parameters) with QLoRA (PEFT method [16] 4.3.3) can be done on 2 A100 80GB GPU. In addition it mitigates the risk of catastrophic forgetting. Typically the dataset for fine-tuning a transformer is much smaller, than the datasets used for pre-training. A full fine-tuning with a small dataset would lead to overfitting, whereas applying a PEFT method reduces the risk of overfitting, through selectively or not updating pretrained parameters [54].

However, fine-tuning a LM offers several challenges. One key challenge is adjusting the hyperparameters for training, such that the model does not overfit or underfit. A model overfits, when it sticks to the fine-tuning data and thus loses creativity for input adjusted responses. Underfitting means a model is to creative and didn't adapt to the training data as much as needed. Optimal hyperparameters for training differ between models. Researchers can transfer their knowledge from fine-tuning in previous works, but newly published models require the experience of researchers and a time consuming and computational expensive trial and error validation [58].

Chapter 3

Related Work

3.1 literature review: ABSA methods

The following literature review primarily draws upon the comprehensive systematic review conducted by Hua et al. (2024) [25] titled 'A Systematic Review of Aspect-based Sentiment Analysis: Domains, Methods, and Trends'. This systematic review provides a robust foundation for understanding the current state of Aspect-based Sentiment Analysis (ABSA), including its domains, methods, and trends. It covers 2 decades of ABSA research with state of the art methodologies up to the 26th of July 2024.

Early ABSA approaches relied heavily on linguistic rules, syntactic features, and lexicon resources. A common technique is using dependency parsing to identify aspect and opinion term candidates, then applying further rules or lexicon lookups for sentiment classification. While these non-machine learning approaches have declined in popularity, they have persisted as a small but steady portion of ABSA research over time, often combined with other techniques.

Traditional machine learning methods emerged as a major paradigm in ABSA, with Support Vector Machines (SVM), Conditional Random Fields (CRF), and Latent Dirichlet Allocation (LDA) being among the most popular. These approaches typically relied on engineered features and annotated datasets. Even as deep learning has come to dominate the field of ABSA, traditional ML techniques have remained popular and were often used in hybrid approaches.

Deep learning methods have seen a rapid rise in ABSA research since 2017, becoming the dominant paradigm. Recurrent Neural Networks (RNNs), particularly Long

Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM) architectures, have been especially popular. These are frequently combined with attention mechanisms to better capture relevant context. Convolutional Neural Networks (CNNs), Graph Neural Networks (GNN) and Graph Convolutional Networks (GCN) have also been applied, but less commonly than RNNs. While CNNs were seen as a real alternative to sequential models GNNs and GCNs is used for external conceptual knowledge (incorporate additional common knowledge to improve the detection in ABSA domain) and syntactic dependency structures.

More recently, transformer-based models leveraging large pre-trained language models like BERT have become increasingly popular for ABSA tasks. These models excel at capturing contextual information and can be fine-tuned on specific ABSA datasets and tasks in various styles. Graph neural networks and Graph Convolutional Networks have also seen noticable growth since 2020. This indicates a rising effort to integrate relational context into the learning process.

An emerging area of research is the application of large generative language models and in-context learning to ABSA tasks. While still in early stages, some studies have explored zero-shot and few-shot learning with models like GPT-3.5 for ABSA [49], though performance has generally lagged behind fine-tuned models. There is growing interest in leveraging these models generative capabilities for data augmentation and formulating ABSA as a text generation task.

Despite methodological advances, the review highlighted persistent challenges in ABSA research. These include heavy reliance on labeled datasets from limited domains (primarily product and service reviews), difficulties in cross-domain transfer, and concerns about model robustness and generalizability. According to the review future research directions may focus on developing more diverse datasets, improving domain adaptation techniques, exploring the use of generative language models for ABSA and exploring data-efficient approaches for low-resource domains.

In conclusion, ABSA methods have evolved significantly over the past decade, with a clear trend towards deep learning and increasingly sophisticated neural architectures. However, traditional techniques remain relevant and are often integrated into hybrid approaches. As the field continues to advance, addressing issues of dataset bias, domain adaptation, and model robustness will be crucial for developing more generalizable and practical ABSA solutions.

Chapter 4

Methodology

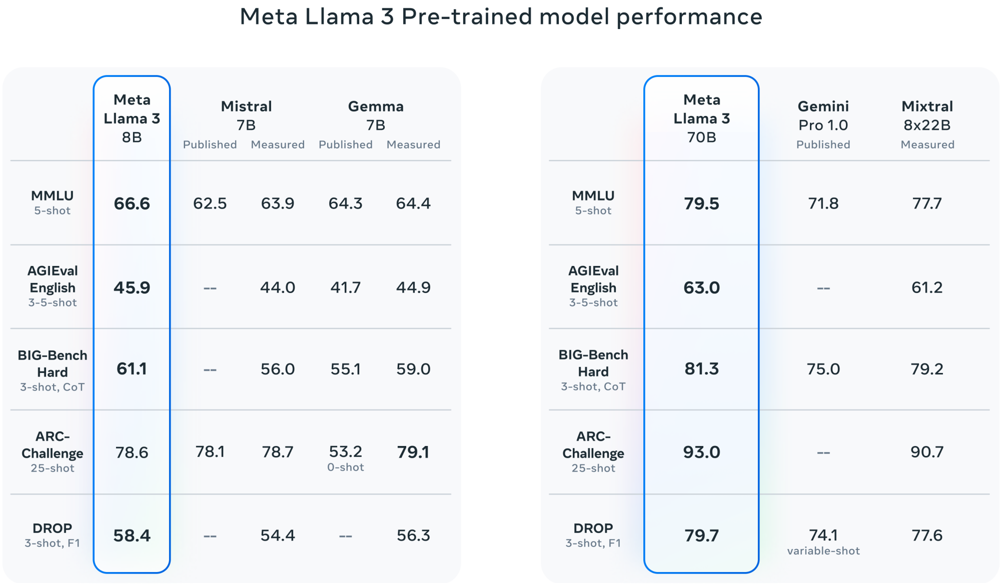
4.1 Introduction to Llama-3 and Phi-2 Models

Llama-3

Released on the 18th April of 2024, the Llama 3[1] model is the newest model of Meta's open-access Large Language Models (LLMs) series. The new model comes in 2 sizes with 8 billion (Llama 3 8B) and 70 billion (Llama 3 70B) parameters, each size with base and instruction-tuned variants. Pretrained on over 15 trillion tokens from publicly available online sources its knowledge goes back to March 2023 for the smaller model and December 2023 for the bigger model. The different variations enable an efficient deployment and fine tuning on consumer-sized GPUs, as well as for more complex tasks on corporate machines. The transformation underlies the principles of auto-regressive language models, which generates text, by predicting the next word in a sequence, based on the words before. Words before can either be a part of the input sequence or next word predictions the model already made. It is optimized for various use cases, particularly dialogue applications, while the owners emphasize better safety and helpfulness [6–8]. They underline this with a recent study 4.1, that shows significant advancements in both pre-training and post-training procedures, which leads to a noticeable decrease in false refusal rates (model falsely refuses to answer due safety), improved responses regarding relevance and accuracy and a wider range of responses from the model. Of particular note is the enhanced functionality of Llama 3 in reasoning, code generation, and following instructions[17]. The creators achieved this by incorporating advanced techniques such as Grouped-Query Attention (GQA) and leveraging massive amounts of high-quality training data. This are key-features to

[1] https://huggingface.co/meta-llama/Meta-Llama-3-8B

us and the reason, why we decided to conduct our research using the llama-3 model. Reasoning is of interest, because the model has to derive implicit ABSA entities, which aren't directly given in the text. We also expect great results from the instruction following talent of the model in our approach for instruction fine-tuning for ABSA [6–8].



Meta Llama 3 Pre-trained model performance

	Meta Llama 3 8B	Mistral 7B		Gemma 7B	
		Published	Measured	Published	Measured
MMLU 5-shot	**66.6**	62.5	63.9	64.3	64.4
AGIEval English 3-5-shot	**45.9**	--	44.0	41.7	44.9
BIG-Bench Hard 3-shot, CoT	**61.1**	--	56.0	55.1	59.0
ARC-Challenge 25-shot	78.6	78.1	78.7	53.2 0-shot	**79.1**
DROP 3-shot, F1	**58.4**	--	54.4	--	56.3

	Meta Llama 3 70B	Gemini Pro 1.0	Mixtral 8x22B
		Published	Measured
MMLU 5-shot	**79.5**	71.8	77.7
AGIEval English 3-5-shot	**63.0**	--	61.2
BIG-Bench Hard 3-shot, CoT	**81.3**	75.0	79.2
ARC-Challenge 25-shot	**93.0**	--	90.7
DROP 3-shot, F1	**79.7**	74.1 variable-shot	77.6

Fig. 4.1 Benchmarks comparing the performance of Llama 3 to other state of the art transformer models with comparable size.
MMLU (Massive Multitask Language Understanding) - 5-shot assesses a model's performance on a wide range of tasks by providing five examples (shots) for context.
AGIEval English - 3-5-shot evaluates a model's performance on English language tasks with 3 to 5 examples for context.
BIG-Bench Hard - 3-shot, CoT (Chain-of-Thought) tests complex reasoning abilities by providing three examples and requiring the model to follow a chain-of-thought process. **ARC-Challenge - 25-shot** measures the model's ability to tackle challenging questions with 25 examples for guidance.
DROP (Discrete Reasoning Over Paragraphs) - 3-shot, F1 focuses on a model's ability to perform discrete reasoning tasks over paragraphs, evaluated with three examples and scored using the F1 metric.
Image from the Meta AI-blog [7].

The Llama architecture represents an advanced transformer-based model designed for natural language processing tasks. It enhances recent innovations in attention mechanisms and normalization techniques to achieve high performance and efficiency in language modeling.

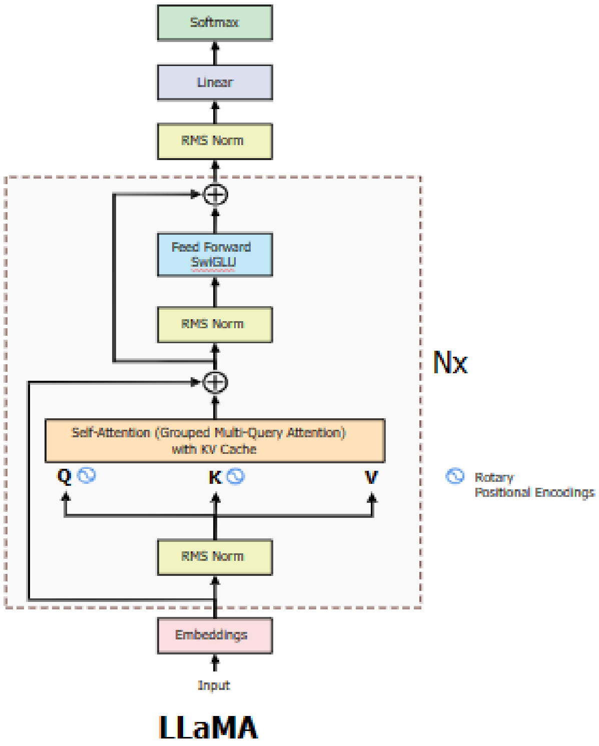


Fig. 4.2 General Llama Design by Meta for model inference

Input Embeddings are tokenized representations of the input text.

RMS Norm (Root Mean Square Layer Normalization stabilizes the input embedding by normalizing them, which helps in maintaining consistent training dynamics and preventing issues related to exploding or vanishing gradients.

Self-Attention Layer (Grouped Multi-Query Attention) allows the model to weigh the importance of different tokens (words (-parts)) in the input sequence dynamically, thereby capturing dependencies and relationships across the sequence.

- The input is transformed into three vectors, Q (Query), K (Key), and V (Value), which are used to calculate attention scores.
- Rotary Positional Encodings help the model considers the order of tokens.
- KV Cache: speeds up processing, reduces overhead at memory access

Feed Forward Network (SwiGLU) enhances non-linear transformations, improving the models ability to learn complex patterns. This network consists of two linear transformations with an activation function in between, enabling more expressive power.

Nx represents the total number of layers in the model. Each layer ensures a more detailed and complex understanding of the input

Linear is a linear transformation, that adjusts the dimensionality of the data to the vocabulary.

Softmax functions convert the linear layer's outputs into probability distributions over the vocabulary, where the model can derive the next probable token (word) from.

Image from vignesh yaadav via his medium article on the Llama architecture [17].

Phi-2

Phi-2[2] is the latest creation of Microsoft. It challenges the approach of creating better language models by scaling it's parameters. With 2.7 billion parameters and a training on it is designed to achieve high performance on various benchmarks, while keeping up with much bigger models, showing outstanding capabilities on reasoning and language understanding capabilities. It even surpasses models that are up to 25x larger, due to a new approach in model scaling and training data curation. The model is trained on 1.4T tokens from multiple passes on a mixture of Synthetic and Web datasets for NLP and coding. The training took 14 days on 96 A100 GPUs [27].

The model is of particular interest, because it comprises a similiar reasoning capability as llama-3 with respect to its size. Its performance and small resource requirements make phi-2 an interesting pick for us in small ABSA applications.

Microsoft earlier published a paper "Textbooks are all you need" [21] introducing phi-1 and it's training data curation. This research is fundamental for the phi model series, underlining the potential of good training data for language models performance. The training data is a mixture of synthetic data created to enhance the models reasoning capabilities and general knowledge along with science, daily activities, daily activities, and theory of mind. In addition the training data consist selected web-data, which is according to Microsoft filtered based on educational value and content quality.

In creation of the phi-2 model, the authors made the choice to implement a concept called Scaled knowledge transfer, where they embedded the knowledge of the 1.3 billion parameters predecessor model Phi-1.5 into the the 2.7 billion parameters model Phi-2. On the one hand, this novel approach accelerates the training convergence. On the other hand it increases the models performance in several benchmark scores 4.3.

[2] https://huggingface.co/microsoft/phi-2

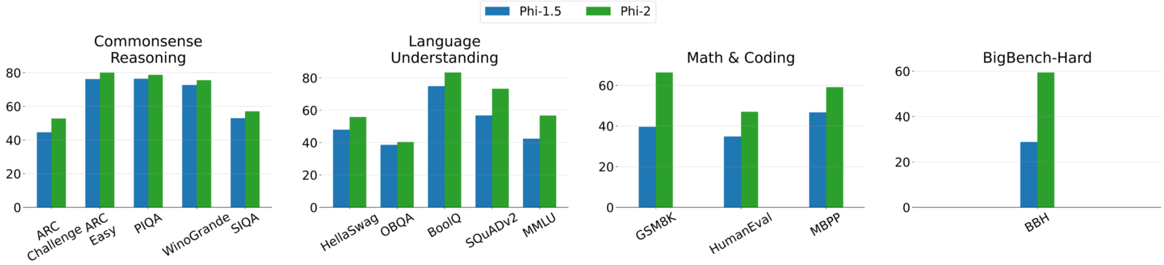
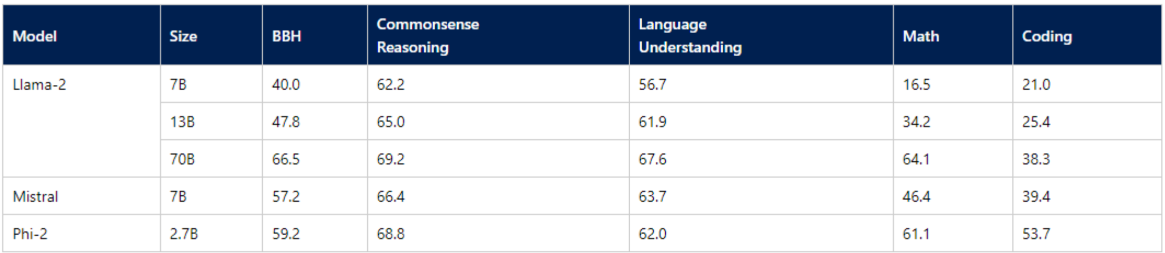


Fig. 4.3 Comparison between Phi-2 (2.7B) and Phi-1.5 (1.3B) models. All tasks are evaluated in 0-shot (no example task given in prompt) except for BBH and MMLU which use 3-shot (multiple examples given, while asking model) CoT and 5-shot, respectively.
Phi-1 (1.3B) achieved superior results in python coding among small language models
Phi-1.5 (1.3B) enhanced capabilities of Phi-1 with focus on common sense reasoning and language understanding (comparable with models 5x larger)
Phi-2 (2.7B) high-end reasoning and language understanding capabilities (comparable with models up to 25x larger)
Source: https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/ [27]



Model	Size	BBH	Commonsense Reasoning	Language Understanding	Math	Coding
Llama-2	7B	40.0	62.2	56.7	16.5	21.0
	13B	47.8	65.0	61.9	34.2	25.4
	70B	66.5	69.2	67.6	64.1	38.3
Mistral	7B	57.2	66.4	63.7	46.4	39.4
Phi-2	2.7B	59.2	68.8	62.0	61.1	53.7

Fig. 4.4 Averaged performance on grouped benchmarks compared to popular open-source small language models. Evaluation is done on Microsoft internal proprietary datasets and tasks.
Source: https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/ [27]

4.2 Dataset Creation

A supervised fine tuning technique is used to fine tune Llama-3 and Phi-2 for ABSA. This approach involves a labeled datasets to train and validate a model. The objective is to fine-tune the models to find aspects and their sentiment polarities of an app review, as well as to categorize each aspect globally. This ABSA methodology is described more in 2.1 and 2.2. To draw further conclusion on the applicability of fine tuned language models for ABSA on app reviews, the models are fine-tuned on a single app dataset [3] and on a dataset multiple containing reviews of multiple apps. The evaluation is conducted on all models and on both datasets as well on a foreign domain dataset of restaurant reviews. The SemEval restaurant dataset, is an open-source gold-standard for ABSA related tasks [39].

The reviews are retrieved from of an Open-RSS-Feed provided by Apple Inc.[4]. Their origin is the US and they are written in english, because the fine-tuned models are pre-trained in english [1, 5]. To provide higher quality reviews, the data is constrained by a filter. Only reviews of the category "mosthelpful" were used. This method allows to retrieve 500 reviews per app at once, resulting in evenly distributed amount of app reviews between apps.

The single app dataset contains reviews from WhatsApp (ID: 310633997)[5]. The multiple app dataset contains reviews from "WhatsApp" (ID: 310633997)[6]5, "Instagram" (ID: 389801252) [7], "Clash of Clans" (ID: 529479190) [8], "Babbel" (ID: 829587759) [9], "Teladoc Health" (ID: 656872607) [10] and "Amazon" (ID: 297606951) [11]. For fair comparison of the fine-tuned models, the multiple apps dataset contains the same WhatsApp reviews.

The multiple apps dataset consists of 2,971 reviews, significantly larger than the single app dataset, which contains 498 reviews. The method of Supervised Fine Tuning involves the prediction of full 'instruct' -instances from our datasets in order to adapt and learn the task. In theory fine-tuning with a large dataset like ours, prevents overfitting of the model and therefore a better generalization on unseen data and domain unspecific tasks. This is because, the model won't be able to memorize all training data, since it's too many for the parameters, we fine-tune. Therefore, we predict that the model understands the task before it learns the labels of our training data.
The distribution of reviews from each app within the larger dataset is approximately equal.

[3] https://huggingface.co/datasets/Alpaca69B/reviews$_{a}ppstore_{w}hatsapp_{a}bsa2$

[4] https://itunes.apple.com/us/rss/customerreviews/page=1/id=310633997/sortBy=mosthelpful/xml

[5] https://apps.apple.com/us/app/whatsapp-messenger/id310633997

[7] https://apps.apple.com/us/app/instagram/id389801252

[8] https://apps.apple.com/us/app/clash-of-clans/id529479190

[9] https://apps.apple.com/us/app/babbel-language-learning/id829587759

[10] https://apps.apple.com/us/app/teladoc-health-telehealth/id656872607

[11] https://apps.apple.com/us/app/amazon-shopping/id297606951

Following standards for machine learning practices, we use Cross Validation and split both the WhatsApp and multiple apps datasets into training, validation, and testing sets with an 80:10:10 ratio. About 80% of the data is used for training, 10% for an evaluation mid training and further 10% is for testing after fine tuning. The SemEval dataset on restuarant reviews and the manual labeled dataset is not split, because it is only used for testing.

The models are fine tuned with a Supervised learning technique, that uses labeled datasets to train and validate models. Therefore a row of a dataset contains the full review with its "title" and "content", as well as the labels "category", "aspect", "sentiment". For training purposes of a text-generation model like Llama-3 and Phi-2 these attributes are embedded into an instruction template. The model is trained on these textual embeddings. This allows the model to mimic the template format and implicitly learn the instruction. Further details on the Fine-Tuning Process are described in 4.3.

The labels for the reviews are retrieved using "Gemini 1.0 Pro" API from Google DeepMind[12][13]. We decided to do synthetic labeling as our corpus of data is too large for manual labeling. A paper from Kai Qiao et al. [43] indicates, that LLMs outperform humans in Aspect-Based Sentiment Analysis labeling tasks, underlining the effectiveness of our approach.

The prompt 5.4 for Gemini is the result of prompt engineering with an observational try and error analysis. This prompt produced the most consistent output for each review in terms of formatting, as seen in 4.7. If the model fails to follow the format shown in 4.7, the output will be pruned. Due to the hard coding nature of the label extraction from the Gemini output, other formats are not accepted.

[12] https://deepmind.google/technologies/gemini/

[13] https://aistudio.google.com/

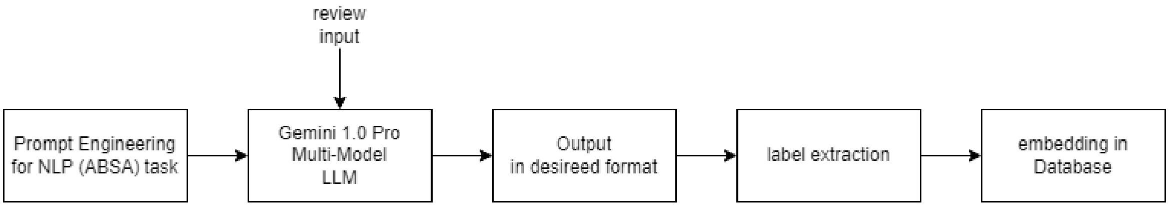

review input
Prompt Engineering for NLP (ABSA) task
Gemini 1.0 Pro Multi-Model LLM
Output in desireed format
label extraction
embedding in Database

Fig. 4.5 Gemini for synthesized Data Annotation:

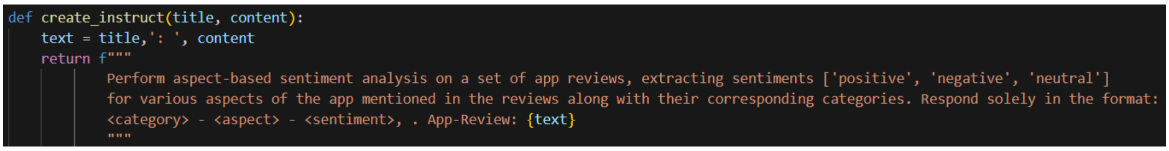
Prompt Engineering for NLP (ABSA) task: Creating and refining prompts to effectively instruct the Gemini for Aspect-Based Sentiment Analysis (ABSA) tasks.

Gemini 1.0 Pro Multi-Model LLM: The refined prompts are then processed by the Gemini model.

Output in desired format: The model returns responses based on the review and the prompt.

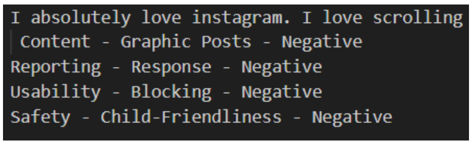
Label extraction: Found labels are extracted from the response (minimal flexibility).

Embedding in Database: Extracted labels and associated data are then embedded into a dataframe for further storage, retrieval, and processing.



```
def create_instruct(title, content):
    text = title,': ', content
    return f"""
            Perform aspect-based sentiment analysis on a set of app reviews, extracting sentiments ['positive', 'negative', 'neutral']
            for various aspects of the app mentioned in the reviews along with their corresponding categories. Respond solely in the format:
            <category> - <aspect> - <sentiment>, . App-Review: {text}
            """
```

Fig. 4.6 ABSA-prompt to extract labels (categories, apects, sentiments) from reviews.



```
I absolutely love instagram. I love scrolling
 Content - Graphic Posts - Negative
Reporting - Response - Negative
Usability - Blocking - Negative
Safety - Child-Friendliness - Negative
```

Fig. 4.7 This example of the gemini output for a prompt shows the accepted format. format structure: category - aspect - sentiment

Although this approach to label the app reviews sounds promising, we acknowledge other research on different domains revealing, that synthetic data creation can decrease the performance of subjective classification tasks [31]. Therefore, we carefully supervised our reviews afterwards and created another dataset, which contains manual labeled reviews from all scraped apps. We choose 6 reviews from each of our 6 apps datasets randomly for manual labeling. We decided to use this manual labeled dataset only for evaluation, because we believe a proper training of the models with it, would lead to an overfitting model. This means, that the model would not be able to generalize well.

To evaluate, how each model performs on the same task but in a different domain, an ABSA dataset on restaurant reviews is utilized. The dataset is a creation of the International Workshop on Semantic Evaluation. The conference was held by North American Chapter of the Association for Computational Linguistics in 2016 Denver, Colorado. Later is was published Open-Source on the website[14] of the Arabic Language Technology Group, which also attended the conference in 2016. It is human annotated, with 163 submissions from 32 teams.

We pre-processed the dataset, meaning formating and filtering. Our ABSA task involve indentifying the sentiments positive, neutral and negative. The SemEval restaurant dataset comprises a 'conflict' sentiment in addition. The initial format, allowed us to derive, which sentences of a review are classified as a 'conflict' and filter these sentences.[15]

4.3 Instruction fine-tuning

Instruction tuning is a novel technique to improve the performance of transformers on diverse natural language processing instructions. As articulated in the famous Google Research paper "Fine-tuned Language Models are Zero-Shot Learners"[53], this approach usually involves the fine-tuning of language models on a collection of tasks specified through instruction templates or natural language instructions. Instruction tuned models have shown significant improvements in zero-shot and few-shot (no or few examples provided within the prompt) learning scenarios. The models were able to transfer its learning and generalize on new unseen data. [53, 56]

An instruction template for fine-tuning usually comprises three elements. For us these are:

- **An Instruction**: A natural language description, which specifies the task. For ABSA fine-tuning, we specify the ABSA task here.
- **Input for Context**: This section is usually additional. It is used to provide context relevant for the task. As we need a text to perform ABSA on, we embed the review (title+content) here.

[14] https://alt.qcri.org/semeval2016/task5/
[15] SemEval2016Task5

- **Output**: The target output for the given prompt, which will be ABSA triplets of category, aspect, sentiment.

instruction = "Instruction: Extract triplets of category - aspect - sentiment, from the given Input. Each pair should include the category, the specific aspect within that category, and the sentiment expressed towards that aspect. Input: review title, review content Output: triplets of category - aspect - sentiment $$$"

Fig. 4.8 Instruction format, which embeds the review and analysis. The natural language instruction is fixed.

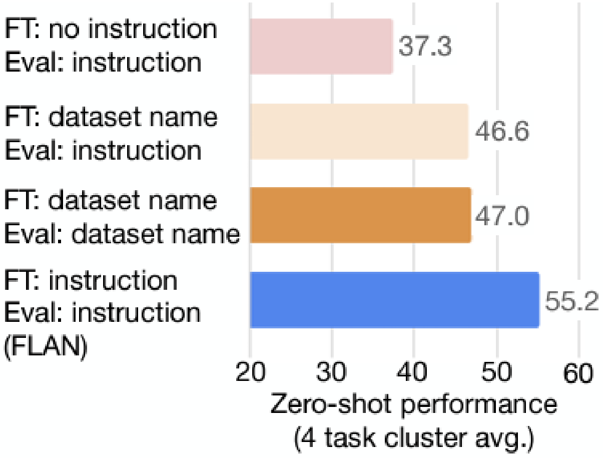


Fig. 4.9 Ablation study result using models with instructions removed from fine-tuning (FT). The results indicate the significance of fine-tuning with natural instructions. source: FLAN paper [53]

We follow current research closely. Thus, our training data is embedded in an instruction template. The strict template explicitly guides the model through a learned instruction. According to the FLAN paper [53] a strict structure is essential for instruction tuning. The template is derived from datasets used in the FLAN study [53]. A recent study on symbol tuning indicates providing natural instructions aren't advantageous, when the fine-tuned model is small or the task is too complex. However, because the study doesn't indicate which models should incorporate natural instructions and which don't, we stick to the recommendation of other studies [53, 56]
After fine-tuning, we observe promising results using our instruction template. In inference the text-generation model completes the template most of the time. Comparable approaches from the FLAN paper [53] and a survey on instruction tuning [61] show promising results in downstreaming tasks, supporting our instruction tuning concept.

In the process of fine-tuning the model is trained in a supervised manner on the instruct column of our datasets. The labels are a shifted representation of the input. This enables the model to learn task-specific data, that enhances the models capabilities and refines its knowledge within the domain of the training data. However, if not fine-tuned correctly, this can lead to overfitting, where the model fails to generalize well due catastrophic forgetting [44].

As part of our research we aim to analyze, how well our a fine-tuned models generalize to unseen data and domains. To mitigate overfitting, we employ several strategies, including hyperparameter tuning 4.3.5 with QLoRA Adapters to find the optimal parameters for fine-tuning of the model. We assess these hyperparameters by constantly evaluating the model during fine-tuning on the evaluation dataset. This shifts the goal from optimizing for the train_loss (difference of prediction to model input (training data example)) to optimizing for the eval_loss (optimizing the loss on unseen data, while training on another dataset).

Theoretically a model is able to generate endless amounts of text. by predicting the next word from the prior words. We flag the end of an analysis in fine-tuning with open brackets. In inference we will only focus on the analysis before the first encounter of these brackets.

4.3.1 Hardware for Fine Tuning

We conduct our research on a single A100-GPU with 40GB of VRAM and an Intel(R) Xeon(R) CPU @ 2.00GHz.

4.3.2 trainable parameters

The phi-2 model consists of a total of 2,779,683,840 parameters. It includes an embedding layer, 32 Decoder layers, and a final linear language modeling head. Each Decoder layer integrates a self-attention mechanism, featuring query, key, and value projection layers (q_{proj}, k_{proj}, v_{proj}), along with an output projection (dense). Additionally, each layer has a multi-layer perceptron (MLP) with two fully connected layers ($fc1$ and $fc2$) and an activation function, alongside layer normalization and dropout components.For the full parameter structure of phi-2 see D.2.

Applying PEFT methods allows us to target a subset of parameters. We choose those in thelinear $Wqkv$, out_{proj}, $fc1$, and $fc2$ modules. These components span the attention and MLP submodules, focusing on key operations like the query, key, value projections, output projection from attention, and the two fully connected layers in the MLP. Out of the 2,779,683,840 total parameters, 2,517,056,000 parameters are actively fine-tuned, corresponding to the selected modules. This represents approximately 90.05% of the entire model's parameter space. For the full parameter structure of phi-2 see D.1.

1. Parameters per Layer:

$$\begin{aligned}
\text{Wqkv (3 projections)} &= 2560 \times 2560 + 2560 = 6,553,600 \\
3 \times 6,553,600 &= 19,660,800 \text{parameters} \\
\text{out\_proj} &= 2560 \times 2560 + 2560 = 6,553,600 \\
\text{fc1} &= 2560 \times 10240 + 10240 = 26,234,240 \\
\text{fc2} &= 10240 \times 2560 + 2560 = 26,239,360 \\
\text{Total per layer} &= 19,660,800 + 6,553,600 + 26,234,240 + 26,239,360 \\
&= 78,687,040 \text{parameters per layer}
\end{aligned}$$

2. Total Parameters Across 32 Layers:

$$\text{Total targeted parameters} = 78,687,040 \times 32 = 2,518,785,280$$

The LLaMA-3 model consists of a total of 8,030,261,248 parameters, including an embedding layer, 32 Decoder layers, and a final linear language modeling head. Each decoder layer features a self-attention mechanism with query (q_{proj}), key (k_{proj}), output projection (o_{proj}) and value (v_{proj}) projection layers. The MLP submodule has three key components: a gating projection ($gate_{proj}$), an up-projection (up_{proj}), and a down-projection ($down_{proj}$), with the activation function SiLU replacing the GELU used in phi-2. Layer normalization is achieved through RMSNorm.

When fine-tuning LLaMA-3 using PEFT methods, we similarly focus on a subset of parameters, including the $Wqkv$ (which includes the query, key, and value projections), o_{proj} (output projection), and the MLP's $gate_{proj}$, up_{proj}, and $down_{proj}$. This fine-tuning targets crucial operations within both the attention mechanism and the MLP submodules. Out of the total 8,030,261,248 parameters, fine-tuning focuses on 7,004,536,832 params, which is about 87,22%.

1. Parameters per Layer:

$$\begin{aligned}\text{Parameters per layer} &= 4096 \times 4096\,(\text{q\_proj}) + 4096 \times 1024\,(\text{k\_proj}) \\ &+ 4096 \times 1024\,(\text{v\_proj}) + 4096 \times 4096\,(\text{o\_proj}) \\ &+ 4096 \times 14336\,(\text{gate\_proj}) + 4096 \times 14336\,(\text{up\_proj}) \\ &+ 14336 \times 4096\,(\text{down\_proj}) \\ &= 16,777,216 + 4,194,304 + 4,194,304 + 16,777,216 \\ &+ 58,982,912 + 58,982,912 + 58,982,912 \\ &= 218,891,776\,\text{parameters per layer}\end{aligned}$$

2. Total Parameters Across 32 Layers:

$$\text{Total targeted parameters} = 218,891,776 \times 32 = 7,004,536,832$$

4.3.3 PEFT

Traditional fine-tuning methods are computationally expensive, because all model parameters have to be readjusted. Commonly known models like GPT-3 span 175 billion parameters. Parameter-efficient fine-tuning techniques (PEFT) 4.3.3 like LoRA [22] and QLoRA [16] adress this problem, reducing the cost of fine-tuning models like GPT-3 significantly [22]. QLoRA is a fine-tuning method that extends the LoRA (Low-Rank-Adapters) method by quantization. We use QLoRA to save on computational cost for training of llama-3 and phi-2 for ABSA. This makes training on the A100 more efficiently, thus saving cost.

In order to understand QLoRA, we will first cover the predecessor method LoRA, which is essentially QLoRA, but without quantization techniques.

Fundamentally **LoRA (Low-Rank Adaptation)** implements trainable low-rank matrices (adapter) into the layers of a pre-trained transformer model, while freezing the original weights. This reduces the number of trainable parameters for fine-tuning.

Consider a pre-trained weight matrix $W \in \mathbb{R}^{d\times k}$ in the transformer model. While in full fine-tuning the matrix W is updated by the accumulated gradient update matrix $\Delta W \in \mathbb{R}^{d\times k}$ directly, in LoRA the weight update $\Delta W \in \mathbb{R}^{d\times k}$ is represented by a product of of two low-rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$. The rank r of the

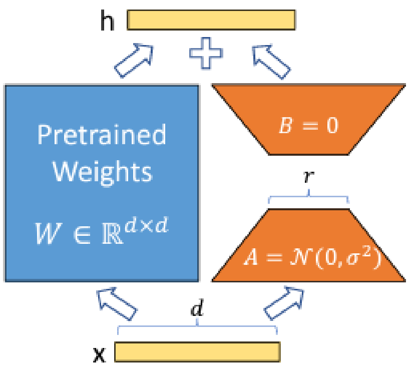

h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Fig. 4.10 LoRA decomposition of weight matrix into pretrained weights and adapter source: LoRA paper [22]

decomposition is smaller than d and k ($r \ll \min(d,k)$). The update of the matrix W can be expressed as:

$$W' = W + \Delta W = W + AB$$

This method reduces the number of trainable parameters from $d \times k \to r \times (d+k)$. The new forward pass looks like $h = Wx + \Delta Wx = Wx + BAx$.[22] In transformers this technique can be applied to any linear layer (subset of weight matrices). The authors suggest using it on W_q, W_k, W_v, W_o, which are the query/key/value/output projection matrices in the self-attention module. For the query and value projection matrices the LoRA adaption looks like:

W$'_q = W_q + A_q B_q$ and $W'_v = W_v + A_v B_v$ The result of the fine-tuning process are the adapters containing AB. The update weights take minimal storage. Merging them with the original weights of the base model results in a new model. This process is low on computational cost.

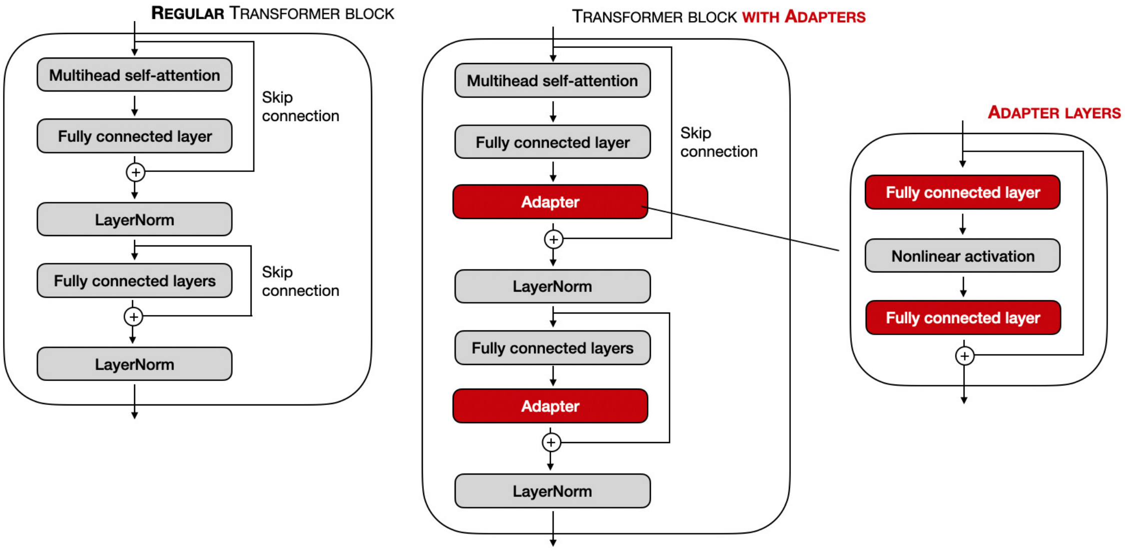
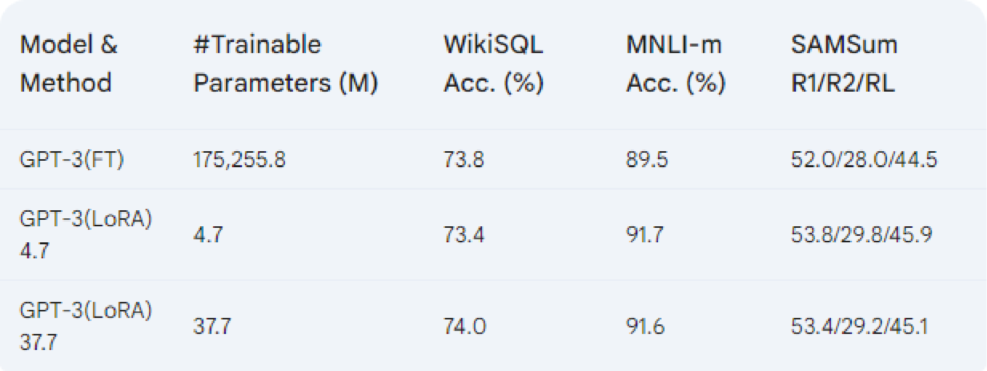


Fig. 4.11 Integration of Adapters in LM for LoRA training and inference of base model with LoRA adapters
source: https://magazine.sebastianraschka.com/p/finetuning-llms-with-adapters

LoRA provides several advantages:

- reduces memory consumption in fine-tuning process and storage consumption for LoRA adapters since $r \ll \min(d,k)$
- updating selective layers limits the risk of catastrophic forgetting (model forgets all pre-trained information)
- preservation of the original weights
- comparable performance to fully fine-tuned models in many cases
- no additional latency as adapters can be merged with base model

Empirical results in the LoRA paper [22] show, that this method can match or surpass the performance of full-fine tuning in several benchmarks and several other fine-tuning methods, while requiring less trainable parameters.



Model & Method	#Trainable Parameters (M)	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3(FT)	175,255.8	73.8	89.5	52.0/28.0/44.5
GPT-3(LoRA) 4.7	4.7	73.4	91.7	53.8/29.8/45.9
GPT-3(LoRA) 37.7	37.7	74.0	91.6	53.4/29.2/45.1

Fig. 4.12 Comparision of full fine-tuning method with LoRA on GPT-3

- WikiSQL: benchmark for question answering over factual knowledge bases
- MNLI-m: assesses the model's ability to understand natural language entailment
- SAMSum R1/R2/RL: benchmark for abstractive summarization

Source: data from LoRA paper Table 4 [22].

QLoRA (Quantized Low-Rank Adaptation) combines low-rank adaption with quantization. This extended version of the LoRA method reduces memory consumption further during fine-tuning, making fine-tuning fast and efficient[16].

Weights of models can be stored in different datatypes. In our case llama-3 and phi-2 parameters are stored in fp16 datatype (torch.dfloat16)[5, 1]. This can be seen in the $config.json$ of each model on huggingface[16][17]. The bigger the datatypes are, the more precise the models gets. But a higher precision significantly increases memory consumption and training time.
Quantization is a technique that reduces the weight parameters precision from floating-point to lower-bit representations. In fine-tuning with QLoRA quantization doesn't effect the performance of the fine-tuned model significantly 4.14, but it saves immense on memory usage [16].

[16] https://huggingface.co/meta-llama/Meta-Llama-3-8B/blob/main/config.json
[17] https://huggingface.co/microsoft/phi-2/blob/main/config.json

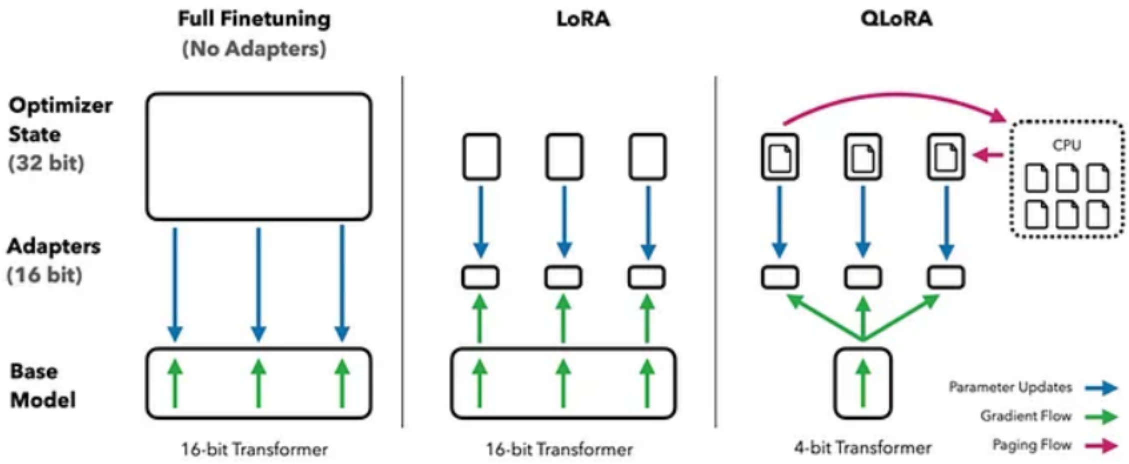
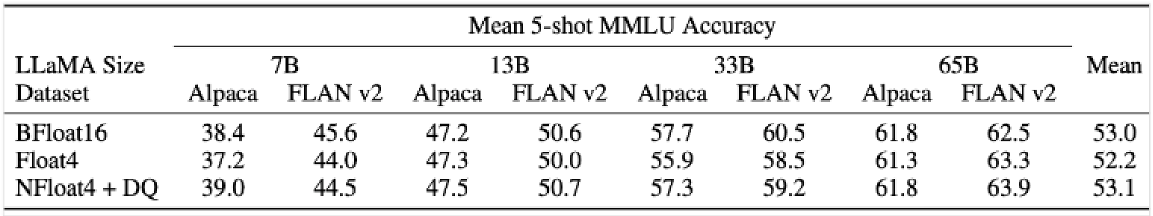


Fig. 4.13 QLoRA improves memory consumption over Full Finetuning and LoRA by quantizing the pre-trained transformer weights
source: QLoRA paper [16]



	Mean 5-shot MMLU Accuracy								
LLaMA Size	7B		13B		33B		65B		Mean
Dataset	Alpaca	FLAN v2	Alpaca	FLAN v2	Alpaca	FLAN v2	Alpaca	FLAN v2	
BFloat16	38.4	45.6	47.2	50.6	57.7	60.5	61.8	62.5	53.0
Float4	37.2	44.0	47.3	50.0	55.9	58.5	61.3	63.3	52.2
NFloat4 + DQ	39.0	44.5	47.5	50.7	57.3	59.2	61.8	63.9	53.1

Fig. 4.14 Perfomance Comparision of full fine-tuning (BFloat16) with QLoRA's memory efficient methods Float4 and NFloat4 + DQ (Double Quantization)
source: QLoRA paper [16]

In QLoRA the base models weights are quantized and then freezed before training 4.13. The paper introduces 3 concepts to manage memory requirements: 4-bit quantization, double quantization, the exploitation of NVIDIA unified memory for paging.

- **4-bit NormalFloat quantization** is an improvement over quantile quantization from T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer from 2022 [15]. It reduces the precision of model weights. The idea is, that the weight distribution follows a normal distribution. By discretizing the continous distribution, this technique ensures an equal number of values in each quantization bin, which minimizes the error of neural network weight quantization.

- **Double Quantization** quantizes the quantization constant used for 4-bit NormalFloat to further reduce memory consumption.

- **Paged Optimizers** manage memory spikes during training. Only for NVIDIA CUDA GPUs, to handle small batches of training data with a high sequence length. Especially helpful, when fine-tuning llm with limited resources.

We use the Paged Optimizer *paged_adamw_32bit* and the 4-bit NormalFloat quantization, which is suggested by the QLoRA paper. Additional memory savings through Double Quantization isn't needed. Not using Double Quantization improves precision, while training.

An optimizer like *paged_adamw_32bit* is an algorithm used to adjust the weights of a model. It tries to minimize the loss function (difference between predicted value and actual outputs).The AdamW is able to set an adaptive learning rate and incorporates momentum for the gradient update. The provided learning rate is thereby a maximum for the optimizer. An optimizer state incorporates the current setting and further data about prior training of the model.

A paged optimizer is beneficial, when maxing out the GPUs VRAM. If the GPU runs out of memory during training, it moves the optimizer states to the CPU RAM and moves them back if needed 4.15. This reduces over-flow errors and has no effect on fine-tuning.

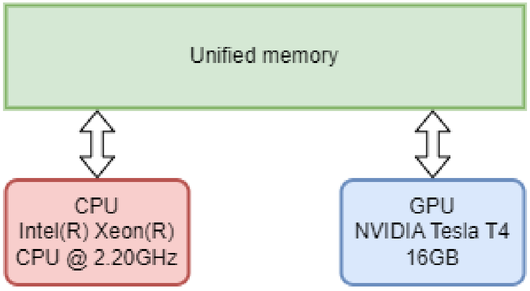
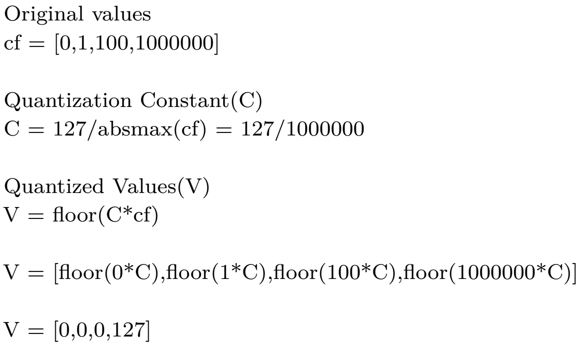


Fig. 4.15 NVIDIA unified memory concept for paged optimizer of QLoRA in training

4-bit NormalFloat quantization is an extension of block-wise quantization to improve the error rate. In block-wise quantization the continuous weight distribution is quantized based on bits. As a result a weight distribution with a high range of values, is quantized to zero values in the low end. This results in catastrophic forgetting 4.16.



Original values
cf = [0,1,100,1000000]

Quantization Constant(C)
C = 127/absmax(cf) = 127/1000000

Quantized Values(V)
V = floor(C*cf)

V = [floor(0*C),floor(1*C),floor(100*C),floor(1000000*C)]

V = [0,0,0,127]

Fig. 4.16 blockwise-quantization with error for high range distribution

Because this approach leads to zeros for weights, we use 4-bit NormalFloat of the QLoRA paper[16]. The idea is to assume, that the weights of a neural network follow the structure of a zero-centered normal distribution with standard derivation σ. The weights of our model are quantized as follows:

1. Assume that the weights follow a normal distribution centered around zero
2. Transform the normal distribution by scaling with σ (using z-score) to fit in a range of [-1,1] (mean = 0, standard deviation = 1)
3. Quantize the distribution in bins with equal many weights. The negative range $[-1,0)$ is divided into 2^{k-1} buckets and the positive $(0,1]$ is divided into $2^{k-1}+1$ bins, where the extra bin for positive values include weights of value zero

In QLoRa the base weights of the base model are quantized 4-bit NormalFloat. During fine-tuning these weights are frozen. The QLoRa adapters follow the original base model dataformat. We load both models in Google's standard dataformat for machine learning models Brain Float (16-bit). For the forward-pass (making predictions) or the backward-pass (updating the weights in training) the frozen quantized weights are dequantized to Brain Float 16-bit to make predictions.

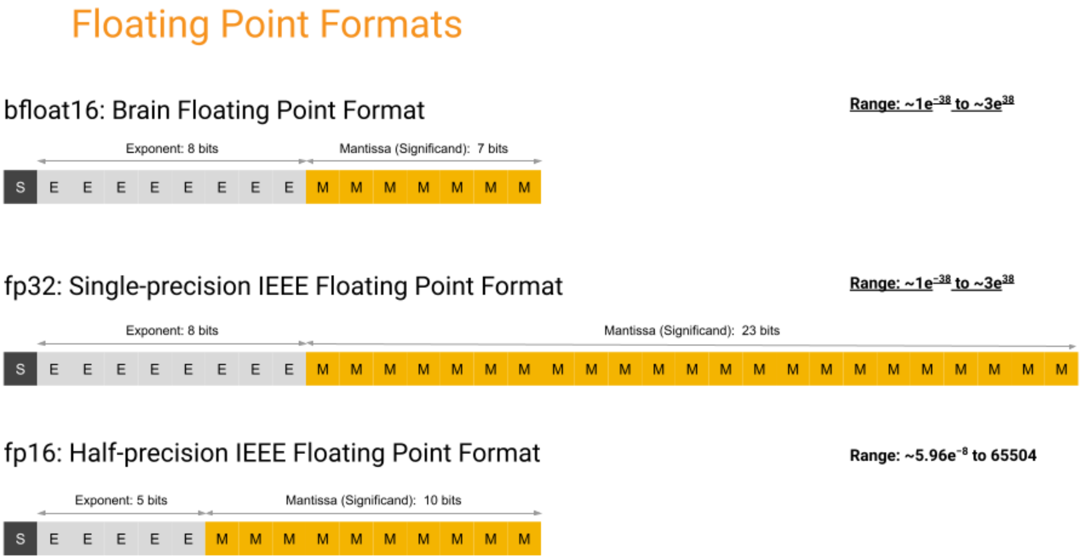

Floating Point Formats
bfloat16: Brain Floating Point Format
Range: ~1e-38 to ~3e38
Exponent: 8 bits
Mantissa (Significand): 7 bits
S E E E E E E E E M M M M M M M
fp32: Single-precision IEEE Floating Point Format
Range: ~1e-38 to ~3e38
Exponent: 8 bits
Mantissa (Significand): 23 bits
S E E E E E E E E M M M M M M M M M M M M M M M M M M M M M M M
fp16: Half-precision IEEE Floating Point Format
Range: ~5.96e-8 to 65504
Exponent: 5 bits
Mantissa (Significand): 10 bits
S E E E E E M M M M M M M M M M

Fig. 4.17 bfloat16 format by Google Brain
bloat16 enables faster training and less memory consumption than bfloat16 and is comparable to float32 in deep neural networks[28]
source: Google: https://cloud.google.com/tpu/docs/bfloat16?hl=de

Brain Float 16 is a different 16-bit floating point format by Google Brain, which enables more efficient computations in terms of speed and memory requirements. When calculating gradient updates differences in values like 1.2345678 and 1.2345679 don't have a big impact on the neuron update, while $1.23x10^5$ compared to $1.23x10^6$ does. The bFloat-16 format provides more bits for the exponent, than for the base. The performance, when computing with bfloat-16 is comparable to float32.[28]

The formular of the QLoRA paper, adapted for our forward-pass within llama-3 8b and phi-2 is defined as follows:

$$Y^{BF16} = X^{BF16} \cdot dequant(c^{FP16}, W^{NF4}) + X^{BF16} \cdot L_1^{BF16} \cdot L_1^{BF16}$$

with $dequant(c^{FP16}, W^{NF4}) = W^{NF4}/c^{FP16} = W^{FP16}$ and conversion to bfloat16

- Y^{BF16}: output prediction of a single layer
- X^{BF16}: input to linear layer in bf16 format
- W^{NF4}: weights of the linear layer, stored in 4-bit NormalFloat Quantization format
- c^{FP16}: scaling constant from quantization process
- L_1^{BF16} and L_2^{BF16}: LoRA Adapters (A and B) in bf16, see 4.10

4.3.4 preprocessing of data for Training

Before starting the training the training data ("Text" column of respective dataset) is cut down to lower computational cost. The "max_seq_length" is a parameter provided for training with the SFTTrainer class from TRL package. It defines the models maximum input length of the "Text" column for training.

The length of input data is measured in tokens. The specific tokenizer is responsible for translating strings into tokens and reverse. Thus, the model can only handle tokens as inputs. The orginal tokenizers from the models are used for (de-)tokenization. Limiting max_seq_length results in a lower number of input neurons for the model, thus a lower number of trainable parameters. A lower maximum input sequence length can thereby be an effective instrument to lower memory cost and training time.

The goal is to filter the training data for an effective maximum sequence length. This resolves issues of memory spikes during training. In order to train both models with the same data, the datasets are filtered the same. A good value for max_seq_length seems to be 500 tokens, which keeps most training data of both datasets.
The change of data distribution due pre-processing can be seen in Appendix E.

4.3.5 Hyperparameter Tuning

The SFTTrainer used for training provides several options for adjusting the training process, including several hyperparameters, that influence the model's training. A hyperparameter can be seen as an variable, that instructs the trainer, how the model should learn the data. Optimal values for these are determined by conducting numerous experiments, with different hyperparameter values, a process called hyperparameter tuning. Finding the optimal values is crucial, because they directly control model's structure, function, and performance.[42]

To find the best hyperparameters for Llama-3 and phi-2, the Optuna framework is utilized. We set the objective of the optimizer to minimizing the *eval_loss*, which is a metric by the trainer, showing how well the model predicts evaluation data. In this case, the evaluation data is from a split of either the "WhatsApp" or the "all apps" dataset. The optimization of the *eval_loss* with hyperparameter tuning reduces the degree of an under- or over-fitting model [50, 59, 34, 10].

Implementing Optuna into our fine-tuning process allows us to conduct an hyperparameter search within for the selected hyperparameters in a set range. This technique operates in studies. Each study involves a full training of the model on the "WhatsApp" or "all apps dataset" dataset. Following the training of each study the *eval_loss* of the model on the evaluation data is reported to Optuna, which then act accordingly and set the hyperparameter values for the next study.

For each model and each dataset 15 studies are conducted, with each of them comprising a different set of values for the hyperparameters.

Several Hyperparameters were tuned:

- **gradient_accumulation_steps**[1 to 6 (int)]:number of steps to accumulate the gradients before updating the model weights.
- **learning_rate**[1e-5 to 1e-4 (float)]: The step size at each iteration while moving towards a minimum of the loss function, determining how quickly the model updates its parameters.
- **num_train_epochs**[1 to 5 (int)]: Bumber of times the model is run on the training data
- **weight_decay**[0.001 to 0.1 (float)]: Prevents overfitting by applying a small penalty to the weights. Prevents exploiding weights. L2-Regularization is typically used.
  $loss = loss + weight\ decay\ parameter \cdot L2\ norm\ of\ the\ weights$
- **lora_alpha**[16 to 128 (int)]: Scaling factor of LoRA layers. Defines the influence of LoRA layers on the models prediction.
- **lora_dropout**[0.05 to 0.2 (float)]: Prevents overfitting in LoRA by randomly setting a portion of the parameters or neurons to zero during training
- **lora_r**[32 to 128 (int)]: Defines the dimension of LoRA matrices used for weight updates. A higher value, allows the model to adapt stronger to complex task. Risk of Overfitting.

We decided to use an hyperparameter optimization approach, because the hyperparameters used in other projects for the models, aren't standardized and differ much.

The different influence of each hyperparameter [A.1, A.2, A.3, A.4] coupled with the different best values for the hyperparameters [A.1,A.2, A.3, A.4] of each model underline the need for hyperparameter tuning.

Hyperparameter tuning of llama-3 on the 'WhatsApp' dataset lowered the $eval_loss$ from ≈ 3.0069 in the worst study to ≈ 1.9784 in the best study. The optimization on the 'all apps' dataset lead to an improvement from ≈ 2.2325 to ≈ 2.1824, reinforcing here the effectiveness of this approach.. The best found hyperparameters for the llama-3 model are listed in tables A.3 ('WhatsApp' dataset) and A.4 ('all apps' dataset).

Hyperparameter tuning of phi-2 on the 'WhatsApp' dataset lowered the $eval_loss$ from ≈ 4 in the worst study to ≈ 2.1491 in the best study. The optimization on the 'all apps' dataset lead to an improvement from ≈ 2.7969 to ≈ 2.0532, reinforcing here the effectiveness of this approach.. The best found hyperparameters for the phi-2 model are listed in tables A.1 ('WhatsApp' dataset) and A.2 ('all apps' dataset).

4.3.6 Model Inference

We mainly use the pipeline class from the transformer package to interfere with a model. The fine-tuned models are text-generation models. We fine-tuned them on a template format with the start keywords " USER:" for the review input and " ASSISTANT:" for the models prediction. As mentioned in 4.3 the model learns this template and completes the request for an analysis following the second keyword.
In Inference we add this template to the review, to advise the model. The *max_length* parameter defines the new maximum number of tokens the model should predict. This limits our analysis length.

```
def generate_prompt(title, text):
full_prompt =f"""### USER: {title}: {text} ### ASSISTANT: """
return full_prompt

pipe = pipeline(task="text-generation", model=model_name,tokenizer = tokenizer,max_length = 500,)
result = pipe(generate_prompt('insane', 'There are some things that could be improved. When you search somethingand use filters, you stil
print(result[0]['generated_text'])
```

Fig. 4.18 Example inference with pipline from transformers package.

4.3.7 Evaluation of Training with TensorBoard

The TensorBoard analysis found in A, shows that our approach to prevent overfitting and underfitting of the model by optimizing for eval_loss is effective. In all models the eval_loss and train_loss decreased over time. The train_loss measure the degree of memorization of the training data and the eval_loss demonstrates the performance on unseen data. An equal decrease of both values indicates that we have chosen effective parameters to train our model on the train dataset and improve general task adaption, while preventing extensive memorization of the training data. Resulting values of both models are have minimal discrepancy, which is a typical signal for good hyperparamter selection.

4.4 Evaluation

The goal is to evaluate how good Llama-3 and Phi-2 work for Aspect-Based Sentiment Analysis on app-data, if they were fine tuned. In addition, we want to know, how domain specific they are, since these models are pre-trained on a wide range of data. There are several metrics used for evaluation machine learning models. The most common metric is a hard evaluation, where the precision, recall and f1 score is measured. Because this metric does not seem fair with an open model, where the set of categories and aspects the model can choose from isn't given, we conducted further performance measuring with a method called LLM as a judge. This approach is inspired by a paper called *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*[66]. We therefore utilized the Gemini-API [18] once more. Gemini works as a judge, evaluation different aspects of the models outputs.
The evaluation is conducted on multiple datasets, including the single app dataset (WhatsApp reviews), the multiple apps dataset (reviews from WhatsApp, Instagram, Clash of Clans, Babbel, Teladoc Health, and Amazon), and an out-of-domain dataset of restaurant reviews from the SemEval 2016 competition. Evaluated are the "test" datasets of WhatsApp, and the multipe apps dataset, which contribute to 10% of their original datasets size (before splitting). The models input in evaluation should not be higher as it is in training, In training "*max_seq_length*" is always set to 500 tokens. Therefore limiting the input length of each test data to 500 tokens in evaluation seems fair.

Multiple evaluations are done, to draw conclusions on the model and domain dependence performance.
Evaluated are:

- WhatsApp fine tuned Phi-2 and Llama-3 evaluated on

[18] https://aistudio.google.com/

  - WhatsApp
  - multiple App dataset
  - SemEval dataset (outside of domain dataset)

- multiple Apps fine tuned Phi-2 and Llama-3 evaluated on
  - WhatsApp
  - multiple App dataset
  - SemEval dataset (outside of domain dataset)

4.4.1 Hard Evaluation

Metrics used for a hard evaluation are precision, recall, and F1-score. These metrics provide a quantitative assessment of the models' ability to correctly identify aspects and their associated sentiments and categories.

- **TP**: Predicted aspect in gold set
- **FP**: : predicted aspect not in gold set
- **FN**: aspect in gold set not in prediction

$$\begin{aligned}\text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}}\\ \text{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}}\\ \text{F1 Score} &= 2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}\end{aligned}$$

Having a bad output, where the output format isn't in the expected format results in an error, but does not effect the precision, recall and f1-score. The format does have to follow the format of the training data, to successfully extract categories, aspects and sentiments. We allow minor deviation by removing spaces at the beginning and additional spaces between categories, aspects and sentiments. In training the character "-" is marks a separation inbetween pairs. If the model falsely starts with "-" it will also be removed. More errors exceed the tolerance of extracting pairs with hard coding and thus result in an error.

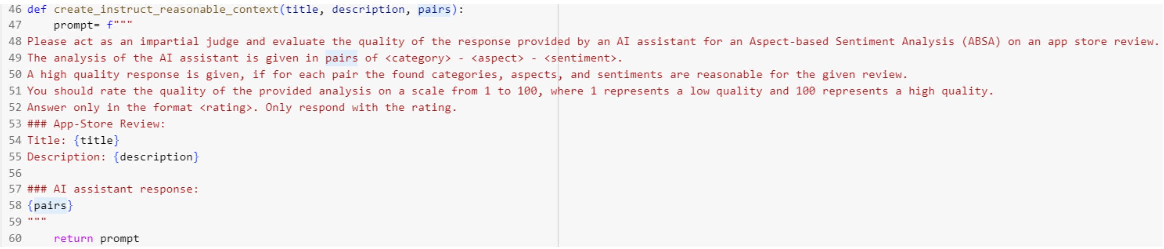
### User: Complain on current and previous update: I love this app yeah cos it’s been basically the commonest app on the app stores with over 7million downloads, I enjoyed the status update option that was introduced too, I’ve been enjoying new features on all updates that’s been brought up until recently when I updated the software And started having issues with searching contacts on my status option, i notices there was another update today so I quickly rushed to get on the latest version and I was surprised and disappointed to see the same problem occurring .. I’m On iOS 14.2 but I don’t think that’s a reason to experience such problems .. please do Something about it as soon as possible.
### Assistant: App - Functionality - Negative)
Predicted Pairs: [(’App’, ’Functionality’, ’Negative’)]
True Pairs: [(’App’, ’Status Update Option’, ’Positive’), (’App’, ’New Features’, ’Positive’), (’App’, ’Searching Contacts’, ’Negative’)]
result_pairs: {’TP’: 0, ’FN’: 3, ’FP’: 1}
result_categories: {’TP’: 1, ’FN’: 0, ’FP’: 0}
result_aspects: {’TP’: 0, ’FN’: 3, ’FP’: 1}
result_sentiments: {’TP’: 1, ’FN’: 2, ’FP’: 0}
result_category_aspect_pair: {’TP’: 0, ’FN’: 3, ’FP’: 1}
result_aspect_sentiment_pair: {’TP’: 0, ’FN’: 3, ’FP’: 1}
result_category_sentiment_pair: {’TP’: 1, ’FN’: 2, ’FP’: 0}

Fig. 4.19 Example for a successful model output and analysis.
result_pairs: evaluates given pairs of category - aspect - sentiment
result_categories: evaluates only categories
result_aspects: evaluates only aspects
result_sentiments: evaluates only sentiments
result_category_aspect_pair: evaluates pairs of category and aspect
result_aspect_sentiment_pair: evaluates only pairs of aspects and sentiments
result_category_sentiment_pair: evaluates only pairs of category and sentiment

We evaluate the success rate of the model as $\frac{number\_of\_tests}{number\_of\_successful\_tests}$. Tests, where the model output allows the extraction of categories, aspects and sentiments is a success. Thus, it doesn’t imply a true prediction. With the findings of a test (example 4.4.1) we further more calculate precision, recall and f1-score for correct categories, aspects, sentiments, triplets (category - aspect - sentiment), pairs (category - aspect, aspect - sentiment, category - sentiment). In addition we count the number of triplets the model outputs and put it in comparison to the triplets in the test dataset.

4.4.2 Soft Evaluation: LLM as a Judge

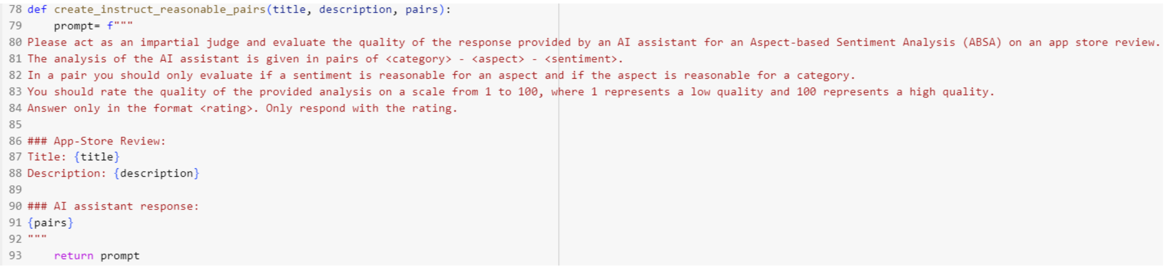
As the answers of our models differ alot with the gold-standard dataset answers, we used a new approach to evaluate our models. A first observation of models outcomes indicate, that the models found labels does not seem to be wrong, although they don’t count as a True Positive. That’s is probably because our fine-tuned language models don’t over- or underfit for the training, thus result in a model with a huge corpus of pre-trained data, where it derives the labels from. By reducing the *training_loss* and



```python
def create_instruct_reasonable_context(title, description, pairs):
    prompt= f"""
Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant for an Aspect-based Sentiment Analysis (ABSA) on an app store review.
The analysis of the AI assistant is given in pairs of <category> - <aspect> - <sentiment>.
A high quality response is given, if for each pair the found categories, aspects, and sentiments are reasonable for the given review.
You should rate the quality of the provided analysis on a scale from 1 to 100, where 1 represents a low quality and 100 represents a high quality.
Answer only in the format <rating>. Only respond with the rating.
### App-Store Review:
Title: {title}
Description: {description}

### AI assistant response:
{pairs}
"""
    return prompt
```

Fig. 4.20

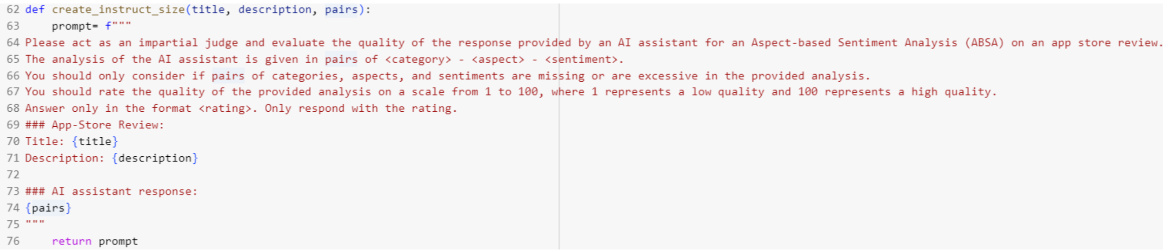


```python
def create_instruct_reasonable_pairs(title, description, pairs):
    prompt= f"""
Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant for an Aspect-based Sentiment Analysis (ABSA) on an app store review.
The analysis of the AI assistant is given in pairs of <category> - <aspect> - <sentiment>.
In a pair you should only evaluate if a sentiment is reasonable for an aspect and if the aspect is reasonable for a category.
You should rate the quality of the provided analysis on a scale from 1 to 100, where 1 represents a low quality and 100 represents a high quality.
Answer only in the format <rating>. Only respond with the rating.

### App-Store Review:
Title: {title}
Description: {description}

### AI assistant response:
{pairs}
"""
    return prompt
```

Fig. 4.21



```python
def create_instruct_size(title, description, pairs):
    prompt= f"""
Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant for an Aspect-based Sentiment Analysis (ABSA) on an app store review.
The analysis of the AI assistant is given in pairs of <category> - <aspect> - <sentiment>.
You should only consider if pairs of categories, aspects, and sentiments are missing or are excessive in the provided analysis.
You should rate the quality of the provided analysis on a scale from 1 to 100, where 1 represents a low quality and 100 represents a high quality.
Answer only in the format <rating>. Only respond with the rating.
### App-Store Review:
Title: {title}
Description: {description}

### AI assistant response:
{pairs}
"""
    return prompt
```

Fig. 4.22

thereby degree of adaption to the training examples, the model would probably stick more to the training examples and limit it's creativity in finding fitting labels. But that would most likely result in a more domain-specific model, where the model could not handle other apps or other domains that well. An underfitting, would have reverse results. Allowing the model more creativity, leads to more unpredictable labels and more error outputs, where the model does not follow the trained format.

By using a LLM as judge, we counter these creative label outputs, which aren't wrong but not a True Positive to the test dataset. We are using Gemini once again to judge our models outputs. We ask Gemini to rate different aspects of the given analysis by our model from 1 to 100. We rate several aspects of the analysis. First we evaluate the models inner structure. Therefore we ask Gemini rate the amount of which a sentiment is reasonable for an aspect and the aspect is reasonable for a category. (4.21) The second aspect we analyze is the amount of the found triplets. There may are missing or excessive triplets in the models output. (4.22) Finally, we check whether the triplets match the context of the input. (4.20)

The extraction of the fine-tuned models triplets is done the same as for the hard evaluation. Minor differences to the wanted format are allowed, else the trial is pruned. In the soft evaluation with Gemini as a judge, we have to extract the answer from gemini as well. We advice Gemini to answer only with the rating. If the response differs, it counts as an error. Following all tests we calculate the success rate of our evaluation. We use the same formular as for the hard evaluation $\frac{number\_of\_tests}{number\_of\_successful\_tests}$.

An analysis of our soft evaluation method is outta scope for this study, but observations in testings are positive. We applied some prompt engineering techniques to adjust our prompts (4.20 4.22 4.21) for better results. Altough there is now paper showing the effectiveness of gemini as a judge, there are several studies indicating a good performance of using gemini similar models. The paper [66] demonstrate, that LLM judges like GPT-4 match human preferences in over 80%. The authors proposed the method LLM-Eval [3] illustrates, that dialogue-optimized LLMs, like Claude and ChatGPT, yield superior performance as judges in evaluating open-domain conversations compared to smaller models. Other paper even indicate a potential to use a llm as a judge for complex tasks like legal reasoning [18, 2].

Chapter 5

Performance Discussion

5.1 Results

We employ both hard and soft evaluation metrics to assess the models complex open ABSA capabilities across various domains, including app reviews and restaurant feedback. Interestingly, while llama-3 slightly outperformed phi-2 in hard evaluation metrics (with summed scores of 1.50 and 1.21, respectively), phi-2 showed marginally better results in soft evaluation. This disparity highlights the importance of employing diverse evaluation methodologies in assessing language model performance.The findings reveal complex patterns of performance and raise important questions about the nature of ABSA tasks and the evaluation of language models in this domain.

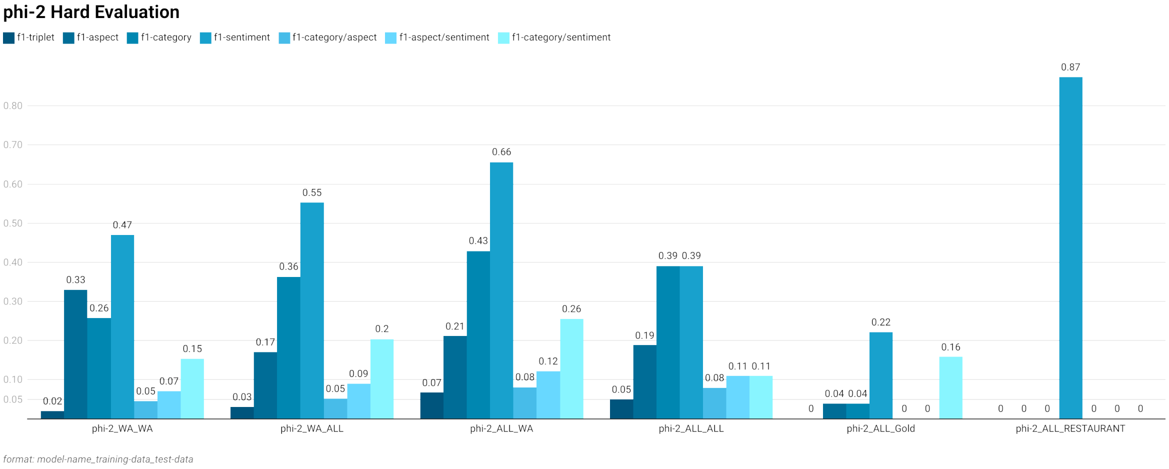
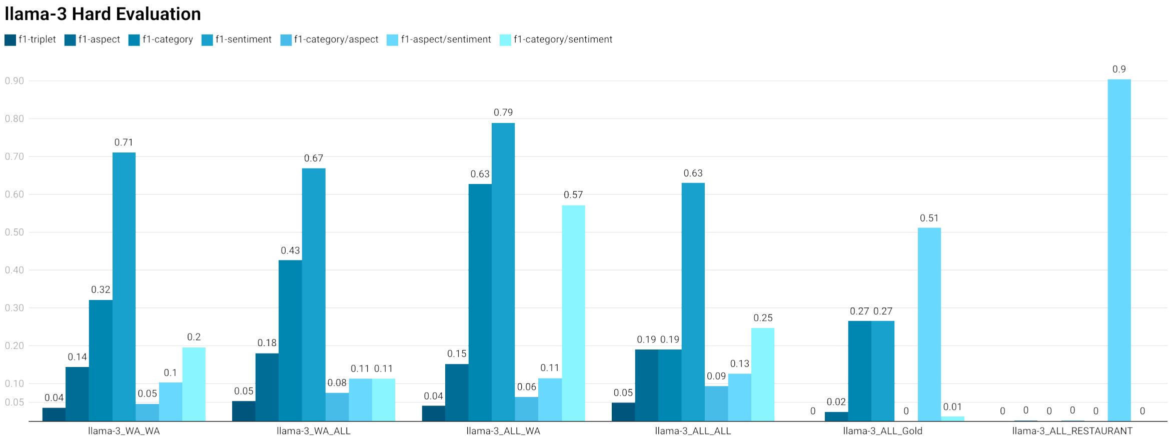


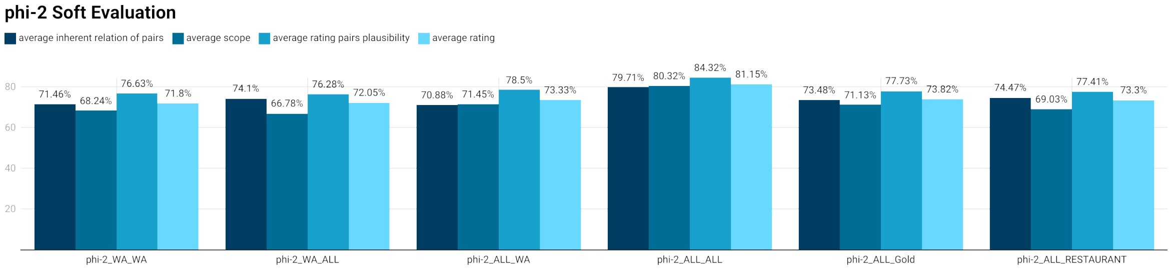
Fig. 5.1



format: model-name_training-data_test-data

Fig. 5.2

All fine-tuned models of llama-3 and phi-2 demonstrated good success rates across all test datasets. The success rate measures the ratio of successful analyses and further processing, which includes the derivation of triplets (category - aspect - sentiment). Observation show no errors with inference of the model. The errors were mostly due formatting of the output. As mentioned earlier, we surpass some mistakes in the derivation of triplets, like unnecessary spaces in the output, but the tolerance is limited. The llama-3 and phi-2 models reached a score of 0.68 and 0.72. Interestingly the success rate is highest among the SemEval restaurant domain dataset with a rate of 0.79 for phi-2 and 0.96. Therefore, we can conclude, that our models have adapted to the task format with a robust domain independency. Furthermore, our observations show, that our models are mostly able to predict triplets, but encounter difficulties with ending there analysis on our makers '$$$'.



format: model-name_training-data_test-data

Fig. 5.3

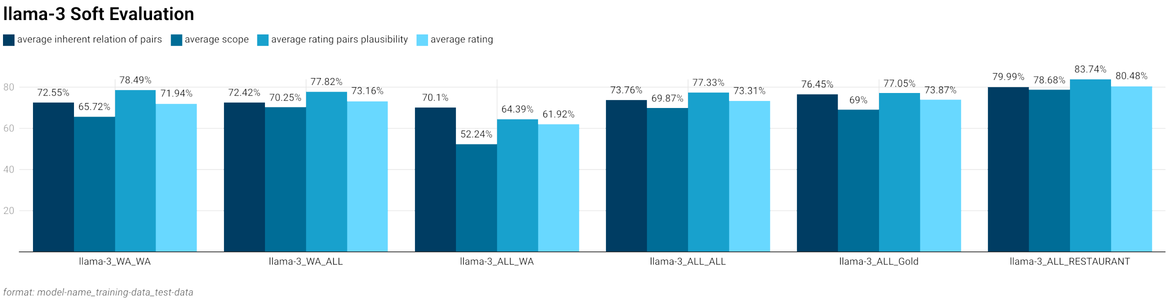


Fig. 5.4

The Soft Evaluation, conducted using Gemini as a judge, provided valuable insights into the models ABSA capabilities. Llama-3 and phi-2 received a high rating (76/100 and 78/100) for 'plausible context aware triplets'. The high score and the low hard evaluation score suggest, that even when predictions deviated from ground truth labels, they often remained contextually appropriate. The relation of categories, aspects, and sentiments was rated equally (74/100) for both models, indicating that the categories and aspects are compatible. Observations noted a tendency towards a hard generalization in category assignment. Surprisingly the scope of the analysis, rating the amount of triplets in regards to the context received similar good ratings, although the hard evaluation shows that all models label more than Gemini or humans (self annotated and SemEval restaurant dataset) would. The Soft Evaluation is helpful indicator to value the conceived labels of our open models, but it didn't confirm our thesis, that the big llama-3 models surpass the performance of the smaller phi-2 models. Parameter size doesn't necessarily result in a better ABSA performance. The performance should be highly effected by the pre-training data. As in 4.1 described the phi-2 base model is highly effective in reasoning tasks. And despite phi-2 being almost 1/3 of the size of llama-3 8b, it goes head to head in these operations. This may be the reason, why phi-2 surpasses the llama-3 model.

A notable finding was the discrepancy between all f1-scores on self-annotated datasets (SELF and SemEval) and those on the original test datasets. This divergence suggests potential differences between human and model labeling patterns, raising questions about the nature of "ground truth" in ABSA tasks. Particularly intriguing was the performance on the SemEval restaurant dataset, where both models achieved low f1-scores but high soft evaluation ratings, surpassing their performance in the app domain. This outcome indicates good domain independence but also suggests that the restaurant domain may present a less challenging ABSA task due to its more common presence in pre-training data.

As teased, the evaluation also revealed significant differences in label prediction quantity between the two models. Llama-3 consistently overpredicted labels, generating up to 2.78 times more triplets than present in the test data for self-annotated datasets. In contrast, phi-2 showed better restraint, but still overpredicted by an average factor of 1.48. Interestingly the bigger model, performed much worse than phi-2 with an average overprediction of factor of 2,26. But this outperformance of the phi-2 model can't be justified by the test on human evaluated datasets. On the SemEval restaurant dataset llama-3 surpassed phi-2. This disparity in prediction behavior warrants further investigation, as it may have implications for the practical application of these models in real-world ABSA tasks.

Both models demonstrated good task comprehension. Phi-2 was able to slightly outperforming llama-3 on task formating(0.72 vs 0.68). The models showed robust domain independence, suprisingly excelling in the restaurant domain. Soft evaluation, conducted using Gemini as a judge, revealed high ratings for both models in producing plausible, context-aware triplets (76/100 for llama-3 and 78/100 for phi-2), even when deviating from ground truth labels. The inherent relation of categories, aspects, and sentiments was rated equally (74/100) for both models. A significant discrepancy was observed between f1-scores on self-annotated datasets and original test datasets, suggesting differences in human and model labeling patterns. Label prediction quantity varied between models, with llama-3 consistently overpredicting (up to 2.78 times more labels than in test data) compared to phi-2's more restrained approach (overpredicting by a factor of 1.48 on average). Interestingly, the study found that a larger model size did not necessarily correlate with better ABSA performance, as evidenced by phi-2's competitive performance despite being smaller than llama-3.
Using multiple apps

5.1.1 Factors Influencing Performance

Several key factors likely influenced the performance of the llama-3 and phi-2 models in Aspect-Based Sentiment Analysis tasks:

Model Architecture: Despite phi-2 being smaller than llama-3, its competitive performance indicates that architectural differences play a crucial role. Phi-2's effectiveness in reasoning tasks, as noted in previous research, may contribute to its strong showing in ABSA, which requires nuanced understanding of context and sentiment.

Fine-tuning Approach: The method used for fine-tuning, including the choice of hyperparameters, learning rate, and the specific fine-tuning dataset, can significantly affect a model's ability to adapt to ABSA tasks. Although, our approach was always to bring training loss and valuation loss in harmony, guaranteeing a theoretical balance between learning the given examples and deriving the task from it to succeed on unknown examples. The observed differences in performance between the models may partially stem from variations in their fine-tuning processes.

Task Complexity: The inherent complexity of ABSA, which requires simultaneous identification of aspects, categories, and sentiments, poses a challenge for language models. Our evaluation results propose, that it may be easier for the model the find the corresponding sentiment to an aspect or category. This thesis can be derived from a high accuracy for the sentiment prediction, but low on aspects and categories, but needs further studies, since our soft evaluation signals good aspect and sentiment prediction.

Domain Specificity: The models performance varied across different domains (app reviews vs. restaurant reviews), highlighting the impact of domain-specific knowledge and terminology on ABSA tasks. The models ability to generalize across domains seems to be influenced by the depth of their pre-training data in relation to these specific domains. Although, the training data, was domain specific and the models performed worse on domains close the training data. Nevertheless, the training effect on complex app domains is questionable due their eligibility for gifting the model descriptive domain knowledge.

5.1.2 Limitations and Challenges

Despite good results in the soft evaluation, a more detailed look on our models reveals several limitations and challenges.

Overprediction Tendency: Both models, particularly llama-3, demonstrated a tendany to overpredict labels, generating more triplets than present in the test data. This overprediction can lead to noise in the analysis and potentially overwhelm users with excessive information.

Inconsistency in Domain Adaptation: While the models showed good domain independence overall, their performance in the varied across different domains. This diverge can be seen in the soft evaluation (Appendix C) on app and SemEval Restaurant reviews. This inconsistency poses challenges in applying these models uniformly across diverse fields without domain-specific knowledge.

Limited Category Granularity: The models often predicted too general categories, lacking the ability to consistently identify more fine-grained domain specific categories. This limitation restricts the depth of analysis for the user.

Divergence from Human Labeling: The discrepancy between model predictions and human-annotated datasets highlights a gap in aligning AI-generated insights with human expert judgments, particularly in nuanced or context-dependent scenarios.

Limited Success Rate: In fine-tuning, we ensured that the models don't overfit by balancing the adaption to the training data and the performance on unseen data. The benefit is, that the models perform better on unseen reviews and show strong task independence. The drawback is less format adaption, which leads to less successful retrievals of ABSA triplets. The models sticked to the format in about 70% of trials.

Evaluation Metric Limitations: The disparity between hard and soft evaluation results underscores the challenge of accurately assessing model performance in open ABSA tasks, where labels truth value isn't binary. Current machine evaluation metrics may not fully capture the nuances of contextually appropriate analysis.

Computational Resource Requirements: The size and complexity of these models, especially llama-3, necessitate substantial computational resources for training and deployment, potentially limiting their accessibility and real-time application. Altough the potential of the smaler phi-2 model provides hope for further advances of small language model capabilities.

Chapter 6

Conclusion and Future Work

6.1 Summary of Findings

This study evaluated the performance of fine-tuned llama-3 and phi-2 models for Aspect-Based Sentiment Analysis (ABSA) across various domains with a specialization on app reviews. Both models demonstrated high task understanding and robust domain independence. The models showed strong capabilities in generating plausible, context-aware sentiment triplets, as evidenced by high soft evaluation scores. However, discrepancies were observed between model predictions and human-annotated datasets, particularly in aspect granularity and label quantity. Both models tended to overpredict labels significantly. Interestingly, model size did not directly correlate with performance, as the smaller phi-2 model competed effectively with the larger llama-3, even surpassing it on multiple tests. These findings underscore the promise of fine-tuned LLMs in ABSA tasks.

6.2 Practical Implications and Applications

Our models demonstrate robust performance across various domains, particularly in app reviews and restaurant feedback, offering versatility that is valuable for clients lacking technical expertise or resources for domain-specific model reconfiguration. However, we warn regarding their performance in highly specialized or expert-level domains not emphasized in pre-training.

The models ability to process large volumes of data is noteworthy, although the current approach is time-intensive, with inference times of 4-7 seconds for a

600-token sequence on an A100 GPU. This time requirement may increase with larger models, potentially impacting scalability for real-time applications.

A key strength of these models is their high contextual understanding, as evidenced by strong soft evaluation scores. This capability is particularly valuable for nuanced sentiment analysis, including scenarios involving sarcasm or detailed feedback within a review.

While not directly tested, the underlying architecture of these LLMs suggests potential for cross-lingual ABSA, a valuable feature for global companies analyzing data in multiple languages.

The structured output format of these models, using triplets and clear analysis endings with '$$$', enables an integration with existing business intelligence tools, enhancing their practical utility in broader data analytics frameworks.

Despite their promising applicability, it's crucial to acknowledge the models' limitations and potential biases to ensure responsible deployment in real-world scenarios.
Additionally, the substantial computational requirements of these large language models make them unsuitable for resource constrained systems such as mobile devices, limiting their applicability in certain scenarios.

6.3 Limitations

The process of training a model is very resource and time consuming hungry. We were limited on computational power, preventing us to analyze bigger models or do a full fine-tuning without PEFT methods.

Doing a hyperparameter-search is underlies several fine-tuning processes. This is very time and hardware consuming. We needed to repeat these fine-tuning and hyperparameter searches several times for our models, because of implementation errors. We optimized the fine-tuning hyperparameters for two models, with 15 studies each, meaning, that we fine-tuned the models after correct implementation 32 times. A bigger study would give more clearance to the importancies and optimal values of the hyperparameters.

Furthermore, despite prior research there are no labeled datasets for app-reviews. We had to create our own datasets for the study. As manual labeling is too time consuming, we had to find another method. We implemented a ’language model as a judge’ method to pre-label the data. In addition, because the quality of the labeling can’t be verified, we manual labeled a split of the dataset as well. This dataset only contains 60 app reviews. A bigger manual labeled app dataset for training and testing would strengthen our findings.

6.4 Future Work

The fine-tuned large language models (LLMs) llama-3 and phi-2 demonstrate significant potential for Aspect-Based Sentiment Analysis (ABSA). Their performance in the Soft Evaluation showed great success. Altough further evaluations with different metrics are needed to further assess the models Open ABSA capabilities.

It is essential to do more research on the evaluation of fine-tuning data for NLP tasks like ABSA. We therefore couldn’t measure the quality of our training data and don’t know the effects on our ABSA models.

It is important to find further metrics or improve existing ones in order to evaluate the quality of an ABSA analysis. In our study the assigned labels of our ABSA models were wrong in the hard evaluation, but scored well in the soft evaluation, underscoring the need for further evaluation methods.

Our study indicate, that language models with good reasoning capabilities like phi-2 perform great on ABSA tasks. How to assess a models quality for ABSA, before fine-tuning still keeps an open question.

The resource requirements are still immense, preventing many users from applying this approach. Quantifying methods to reduce the models size, seem to be a valuable asset for further research.

References

[1] (2023). [Online] https://huggingface.co/microsoft/phi-2.

[2] (2023). 3. large language models as tax attorneys: A case study in legal capabilities emergence.

[3] (2023). Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models.

[4] (2023). Memd-absa: A multi-element multi-domain dataset for aspect-based sentiment analysis.

[5] (2024). [Online] https://huggingface.co/meta-llama/Meta-Llama-3-8B.

[6] (2024a). Build the future of ai with meta llama 3. [Online] https://llama.meta.com/llama3/.

[7] (2024b). Introducing meta llama 3: The most capable openly available llm to date. [Online] https://ai.meta.com/blog/meta-llama-3/.

[8] (2024c). Model details. [Online] https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[9] Akhtar, M., Ekbal, A., and Bhattacharyya, P. (2018). *Aspect Based Sentiment Analysis: Category Detection and Sentiment Classification for Hindi*, pages 246–257.

[10] Ali, Y. A., Awwad, E. M., Al-Razgan, M., and Maarouf, A. (2023). Hyperparameter search for machine learning algorithms for optimizing the computational complexity. *Processes*.

[11] Busst, M. M. A., Anbananthen, K. S. M., Kannan, S., and Kannan, R. (2024). Extracting explicit and implicit aspects using deep learning.

[12] Cambria, E. (2016). Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107.

[13] Chifu, A.-G. and Fournier, S. (2024). Linguistic features for sentence difficulty prediction in absa. *arXiv.org*.

[14] Debbah, M. (2023). Large language models for telecom. *2023 Eighth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 3–4.

[15] Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. (2022). 8-bit optimizers via block-wise quantization.

[16] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms.

[17] et. al., V. (2023). Exploring and building the llama 3 architecture: A deep dive into components, coding, and inference techniques.

[18] Faozan, B. A. (2023). 5. large language models as fiduciaries: A case study toward robustly communicating with artificial intelligence through legal standards.

[19] Ganganwar, V. and Rajalakshmi, R. (2019). Implicit aspect extraction for sentiment analysis: A survey of recent approaches. *Procedia Computer Science*, 165:485–491. 2nd International Conference on Recent Trends in Advanced Computing ICRTAC -DISRUP - TIV INNOVATION , 2019 November 11-12, 2019.

[20] Gao, Y., Gao, Y., and Gao, Y. (2024). Ckg: Improving absa with text augmentation using chatgpt and knowledge-enhanced gated attention graph convolutional networks.

[21] Gunasekar, S., Zhang, Y., Aneja, J., Cesar, C., Mendes, T., Giorno, A. D., Gopi, S., Javaheripi, M., Kauffmann, P., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Singh Behl, H., Wang, X., Bubeck, S., Eldan, R., Kalai, A. T., Lee, Y. T., and Li, Y. (2023). Textbooks are all you need.

[22] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models.

[23] Hua, Y., Denny, P., Taskova, K., and Wicker, J. (2023a). A systematic review of aspect-based sentiment analysis (absa): Domains, methods, and trends. *ArXiv*, abs/2311.10777.

[24] Hua, Y., Denny, P., Taskova, K., and Wicker, J. (2023b). A systematic review of aspect-based sentiment analysis (absa): Domains, methods, and trends. *arXiv.org*.

[25] Hua, Y. C., Denny, P., Taskova, K., and Wicker, J. (2024). A systematic review of aspect-based sentiment analysis: Domains, methods, and trends.

[26] Huda, C., Heryadi, Y., Lukas, and Budiharto, W. (2022). Aspect-based sentiment analysis in tourism industry for tourism recommender system. *2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 407–412.

[27] Javaheripi, M. and Bubeck, S. (2023). Phi-2: The surprising power of small language models. Microsoft Research Blog, accessed: 2024-06-19.

[28] Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D. T., Jammalamadaka, N., Huang, J., Yuen, H., Yang, J., Park, J., Heinecke, A., Georganas, E., Srinivasan, S., Kundu, A., Smelyanskiy, M., Kaul, B., and Dubey, P. (2019). A study of bfloat16 for deep learning training.

[29] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations.

[30] Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation.

[31] Li, Z., Zhu, H., Lu, Z., and Yin, M. (2023). Synthetic data generation with large language models for text classification: Potential and limitations.

[32] Liu, B. (2012). Sentiment analysis and opinion mining. volume 5.

[33] Liu, N. and Shen, B. (2020). Aspect-based sentiment analysis with gated alternate neural network. *Knowl. Based Syst.*, 188.

[34] Liu, X. and Wang, C. (2021). An empirical study on hyperparameter optimization for fine-tuning pre-trained language models.

[35] Maitama, J. Z., Idris, N., Abdi, A., Shuib, L., and Fauzi, R. (2020). A systematic review on implicit and explicit aspect extraction in sentiment analysis. *IEEE Access*, 8:194166–194191.

[36] Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heinz, I., and Roth, D. (2021). Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56:1 – 40.

[37] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Barnes, N., and Mian, A. (2023). A comprehensive overview of large language models. *ArXiv*, abs/2307.06435.

[38] Nazir, A., Rao, Y., Wu, L., and Sun, L. (2022). Issues and challenges of aspect-based sentiment analysis: A comprehensive survey. *IEEE Transactions on Affective Computing*, 13(2):845–863.

[39] of SemEval-2016, O. (2016). Task 5: Aspect-based sentiment analysis - semeval-2016 task 5. [Online] https://alt.qcri.org/semeval2016/task5/.

[40] Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S. M., and Eryiğit, G. (2016). SemEval-2016 task 5: Aspect based sentiment analysis. In Bethard, S., Carpuat, M., Cer, D., Jurgens, D., Nakov, P., and Zesch, T., editors, *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30, San Diego, California. Association for Computational Linguistics.

[41] Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., and Androutsopoulos, I. (2015). SemEval-2015 task 12: Aspect based sentiment analysis. In Nakov, P., Zesch, T., Cer, D., and Jurgens, D., editors, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, Denver, Colorado. Association for Computational Linguistics.

[42] Probst, P., Boulesteix, A., and Bischl, B. (2018). Tunability: Importance of hyperparameters of machine learning algorithms. *J. Mach. Learn. Res.*, 20.

[43] Qiao, K. (2023). Utilizing large language models for the generation of aspect-based sentiment analysis datasets.

[44] Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. (2020). Rapid learning or feature reuse? towards understanding the effectiveness of maml.

[45] Saad, T. B., Ahmed, M., Ahmed, B., and Sazan, S. A. (2024). A novel transformer based deep learning approach of sentiment analysis for movie reviews. In *2024 6th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, pages 1228–1233.

[46] Schouten, K. and Frasincar, F. (2016). Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):813–830.

[47] Sharma, S. and Saraswat, M. (2024). A robust approach for aspect-based sentiment analysis using deep learning and domain ontologies.

[48] Siddiqua, A., Bindumathi, V., Raghu, G., and Bhargav, Y. (2024). Aspect-based sentiment analysis (absa) using machine learning algorithms.

[49] Simmering, P. F. and Huoviala, P. (2023). Large language models for aspect-based sentiment analysis.

[50] Tribes, C., Benarroch-Lelong, S., Lu, P., and Kobyzev, I. (2023). Hyperparameter optimization for large language model instruction-tuning. *arXiv.org*.

[51] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[52] Wan, H., Yang, Y., Du, J., Liu, Y., Qi, K., and Pan, J. Z. (2020). Target-aspect-sentiment joint detection for aspect-based sentiment analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9122–9129.

[53] Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2022). Finetuned language models are zero-shot learners.

[54] Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., and Wang, F. L. (2023a). Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment.

[55] Xu, Q., Zhu, L., Dai, T., Guo, L., and Cao, S. (2020). Non-negative matrix factorization for implicit aspect identification. *Journal of Ambient Intelligence and Humanized Computing*, 11:2683–2699.

[56] Xu, Z., Shen, Y., and Huang, L. (2023b). Multiinstruct: Improving multi-modal zero-shot learning via instruction tuning.

[57] Yang, T., Yin, Q., Yang, L., and Wu, O. (2022). Aspect-based sentiment analysis with new target representation and dependency attention. *IEEE Transactions on Affective Computing*, 13(2):640–650.

[58] Yu, T. and Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications.

[59] Zhan, T. (2022). Hyper-parameter tuning in deep neural network learning. *Artificial Intelligence and Applications*.

[60] Zhang, H., Cheah, Y.-N., Alyasiri, O. M., and An, J. (2024a). Exploring aspect-based sentiment quadruple extraction with implicit aspects, opinions, and chatgpt: a comprehensive survey. *Artif. Intell. Rev.*, 57(2):17.

[61] Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., and Wang, G. (2024b). Instruction tuning for large language models: A survey.

[62] Zhang, W., Li, X., Deng, Y., Bing, L., and Lam, W. (2022). A survey on aspect-based sentiment analysis: Tasks, methods, and challenges.

[63] Zhang, W., Li, X., Deng, Y., Bing, L., and Lam, W. (2023). A survey on aspect-based sentiment analysis: Tasks, methods, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11019–11038.

[64] Zhang, Y., Du, J., Ma, X., Wen, H., and Fortino, G. (2021a). Aspect-based sentiment analysis for user reviews. *Cognitive Computation*, 13:1114 – 1127.

[65] Zhang, Y., Du, J., Ma, X., Wen, H., and Fortino, G. (2021b). Aspect-based sentiment analysis for user reviews. *Cognitive Computation*, 13.

[66] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena.

All URLs were last accessed on Sep 24, 2024.

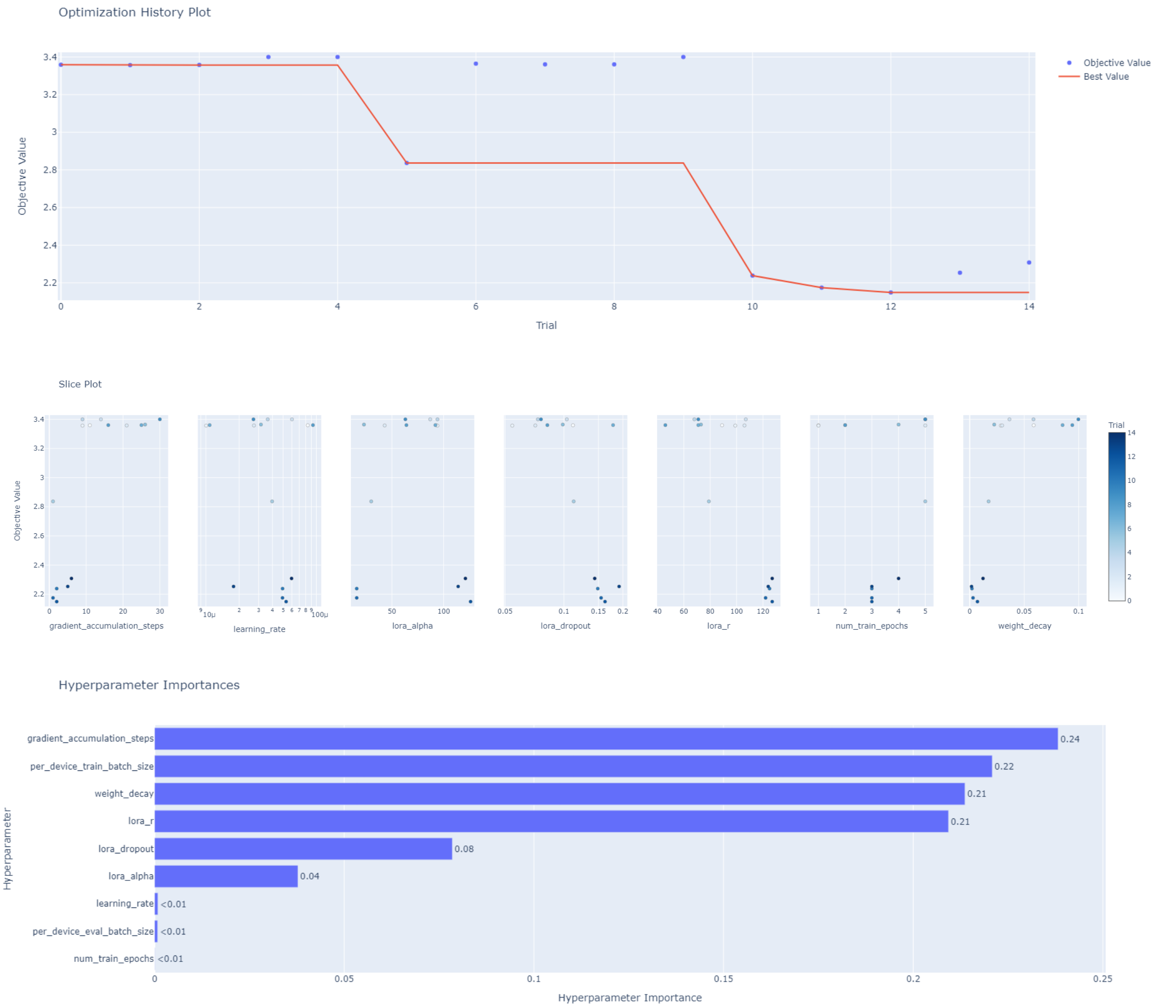
Appendix A

Hyperparameters

This section details the hyperparameters used for experimentation. The optimal parameters of phi-2 and llama-3 are chosen following a hyperparameter opimization. Choosing the best hyperparameters can only be done by experiments and from experience. We set a range for hyperparameter search from experience and conduct 15 studies to find the best model. The optimization took part on llama-3 WhatsApp/all dataset and on phi-2 WhatsApp/all dataset.

The slice-plot shows the error of a model on the validation dataset, when choosing a specific hyperparameter value. This error is further influenced by other hyperparameters. Therefore, there is an error in the hyperparameter importancies, which reduces with further studies.

Fig. A.1 Hyperparameter optimization of phi-2 on the WhatsApp dataset.



Optimization History Plot

Slice Plot

Hyperparameter Importances

Table A.1 Phi-2 trained on 'WhatsApp' dataset best hyperparameters of optimization found in trial 12

Parameter	Value
lora_alpha	126
lora_dropout	0.1621
lora_r	127
gradient_accumulation_steps	2
learning_rate	5.3234e-5
weight_decay	0.007
num_train_epochs	2
per_device_train_batch_size	1
per_device_eval_batch_size	9

Fig. A.2 Hyperparameter optimization of phi-2 on the all apps dataset.

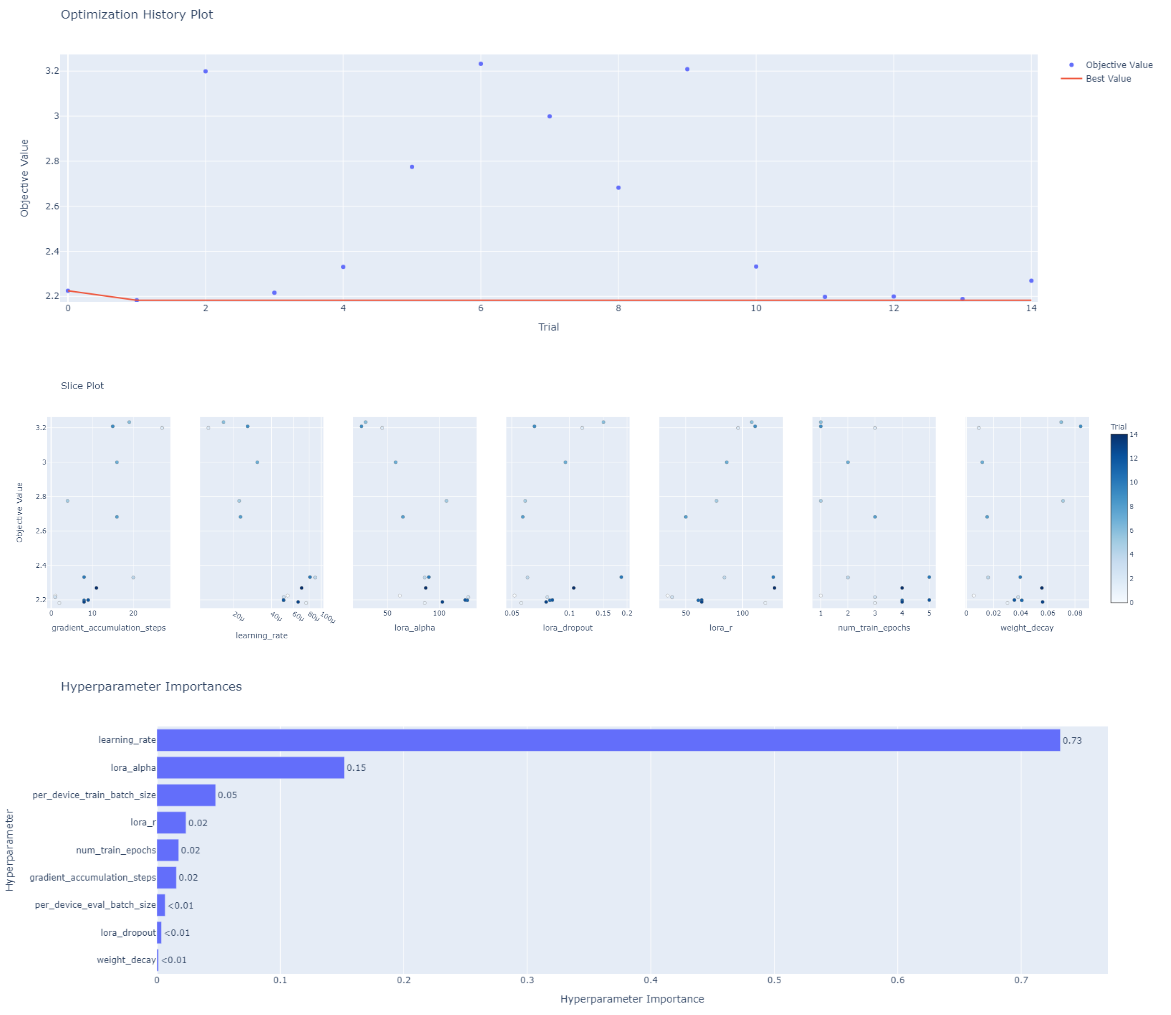


Table A.2 Phi-2 trained on 'all apps' dataset best hyperparameters of optimization found in trial 1

Parameter	Value
lora_alpha	86
lora_dropout	0.0559
lora_r	120
gradient_accumulation_steps	2
learning_rate	7.5898e-5
weight_decay	0.0302
num_train_epochs	2
per_device_train_batch_size	2
per_device_eval_batch_size	11

Fig. A.3 Hyperparameter optimization of llama-3 on the WhatsApp dataset. Trial 4 resulted in an error, because of the intended combination of hyperparameters for this trial and insufficient memory.

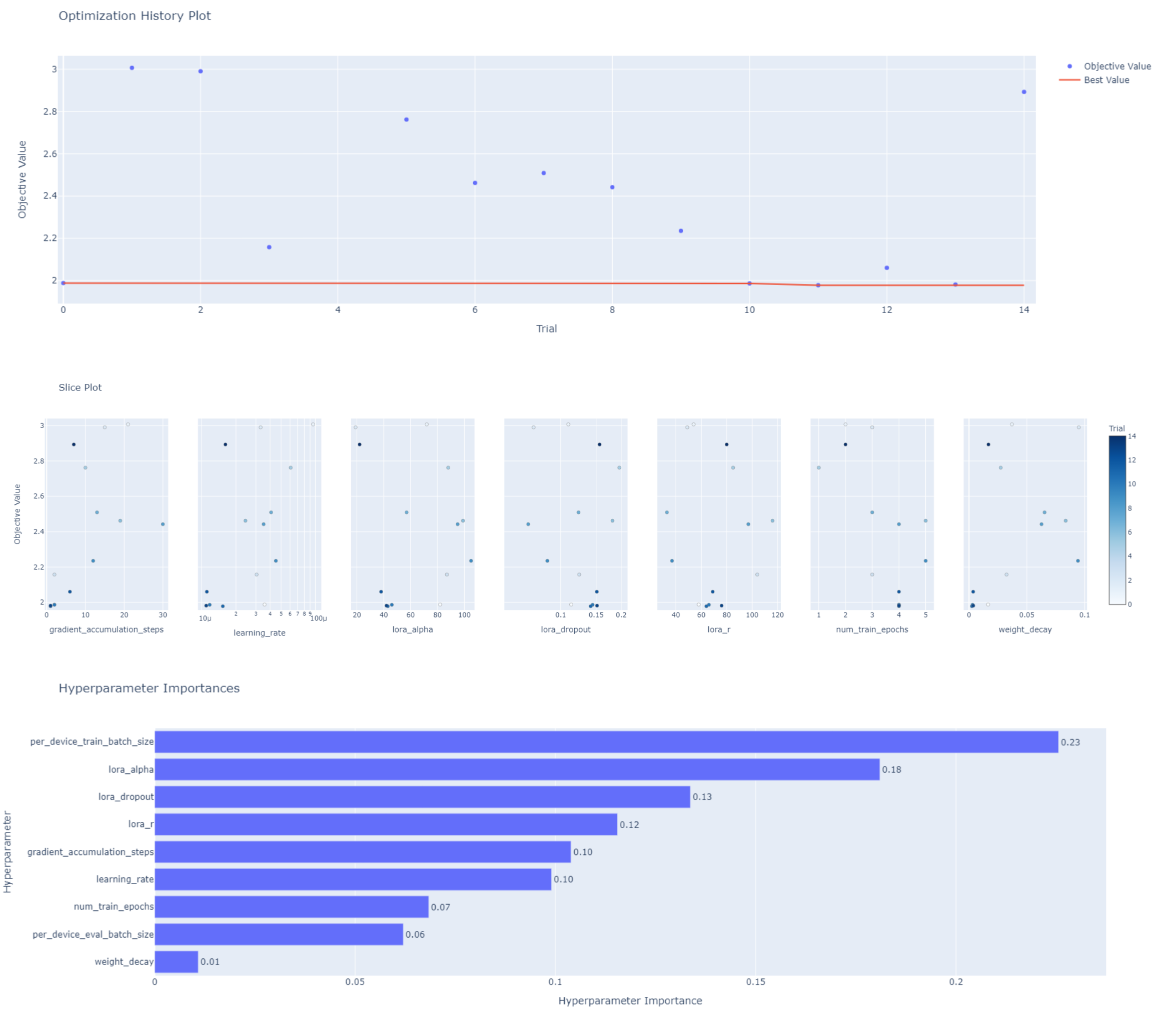


Table A.3 llama-3 trained on ’WhatsApp’ apps dataset best hyperparameters of optimization found in trial 11

Parameter	Value
lora_alpha	43
lora_dropout	0.1409
lora_r	64
gradient_accumulation_steps	1
learning_rate	1.5333e-5
weight_decay	0.0025
num_train_epochs	2
per_device_train_batch_size	1
per_device_eval_batch_size	30

Fig. A.4 Hyperparameter optimization of llama-3 on the all apps dataset. Trial 10 resulted in an error, because of the intended combination of hyperparameters for this trial and insufficient memory.

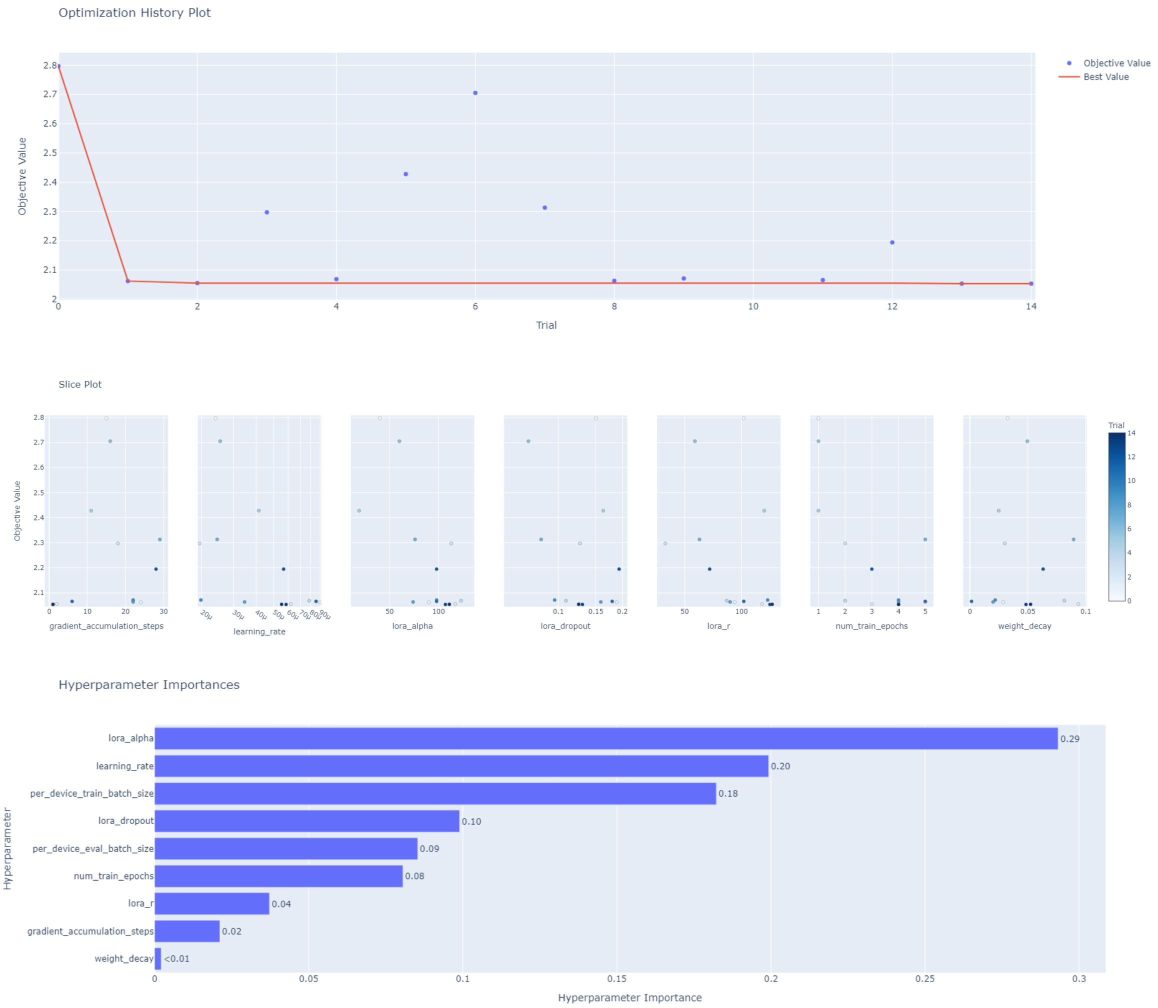
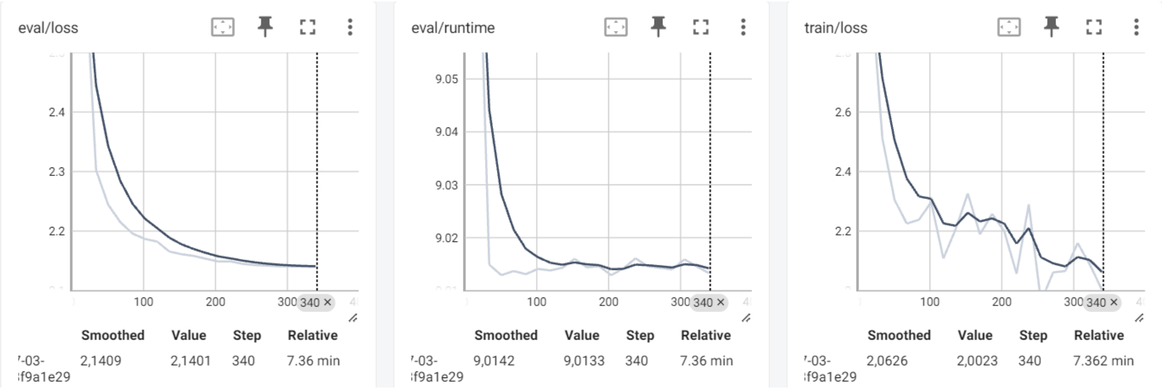


Table A.4 llama-3 trained on 'all apps' dataset best hyperparameters of optimization found in trial 13

Parameter	Value
lora_alpha	107
lora_dropout	0.1297
lora_r	125
gradient_accumulation_steps	1
learning_rate	5.8464e-5
weight_decay	0.0483
num_train_epochs	2
per_device_train_batch_size	21
per_device_eval_batch_size	9

Appendix B

Tensorboard Reports



eval/loss

	Smoothed	Value	Step	Relative
'-03- ;f9a1e29	2,1409	2,1401	340	7.36 min

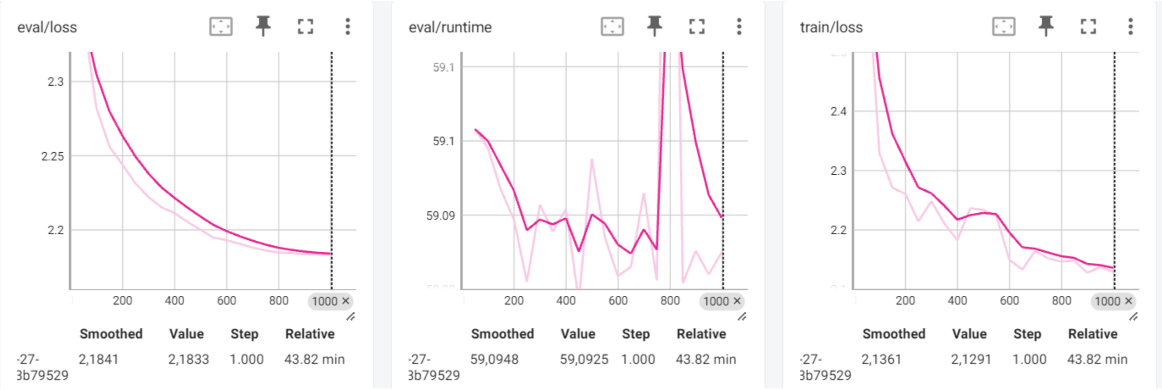
eval/runtime

	Smoothed	Value	Step	Relative
'-03- ;f9a1e29	9,0142	9,0133	340	7.36 min

train/loss

	Smoothed	Value	Step	Relative
'-03- ;f9a1e29	2,0626	2,0023	340	7.362 min

Fig. B.1 Phi-2 trained on WhatsApp dataset



eval/loss

	Smoothed	Value	Step	Relative
-27- 3b79529	2,1841	2,1833	1.000	43.82 min

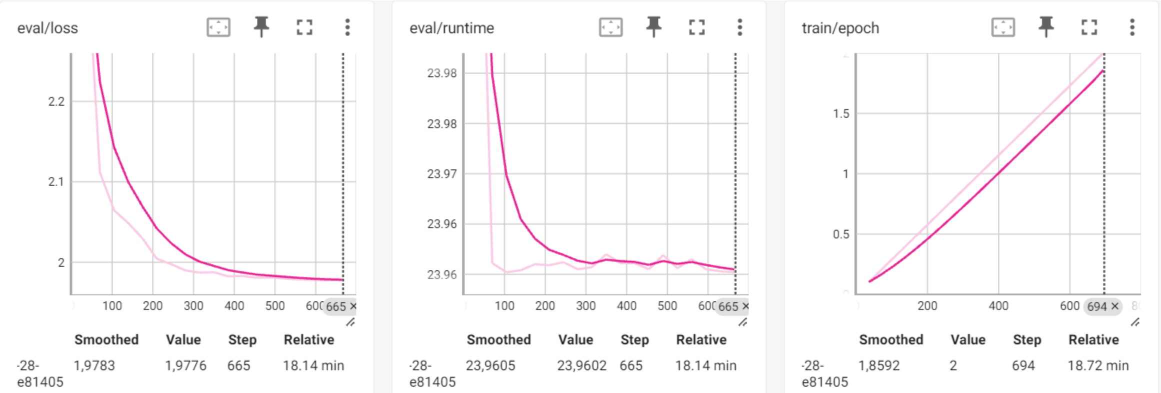
eval/runtime

	Smoothed	Value	Step	Relative
-27- 3b79529	59,0948	59,0925	1.000	43.82 min

train/loss

	Smoothed	Value	Step	Relative
-27- 3b79529	2,1361	2,1291	1.000	43.82 min

Fig. B.2 phi-2 trained on All apps dataset



eval/loss

	Smoothed	Value	Step	Relative
-28- e81405	1,9783	1,9776	665	18.14 min

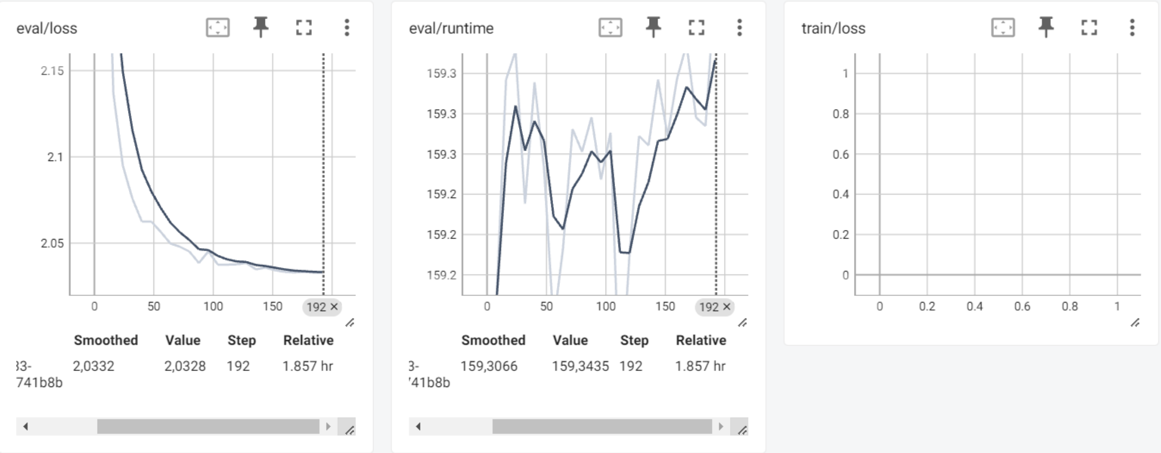
eval/runtime

	Smoothed	Value	Step	Relative
-28- e81405	23,9605	23,9602	665	18.14 min

train/epoch

	Smoothed	Value	Step	Relative
-28- e81405	1,8592	2	694	18.72 min

Fig. B.3 Llama-3 trained on WhatsApp dataset



eval/loss

	Smoothed	Value	Step	Relative
3- 741b8b	2,0332	2,0328	192	1.857 hr

eval/runtime

	Smoothed	Value	Step	Relative
3- '41b8b	159,3066	159,3435	192	1.857 hr

Fig. B.4 Llama-3 trained on All apps dataset

Appendix C

Evaluation Results

Hard Evaluation	phi-2 WA WA	phi-2 WA ALL	phi-2 ALL WA	phi-2 ALL ALL	phi-2 ALL SELF	phi-2 ALL SEMEVAL	average
number of tests	74	446	74	446	60	86	
number of successful tests	50	307	50	290	37	68	
number of error	24	139	24	156	23	18	
success rate	0.676	0.688	0.676	0.650	0.617	0.791	0.683
f1-triplet	0.019	0.030	0.067	0.049	0.000	0.000	0.028
f1-aspect	0.329	0.171	0.212	0.188	0.039	0.000	0.157
f1-category	0.258	0.363	0.429	0.391	0.039	0.000	0.247
f1-sentiment	0.470	0.553	0.656	0.391	0.221	0.873	0.527
f1-category/aspect	0.045	0.052	0.081	0.079	0.000	0.000	0.043
f1-aspect/sentiment	0.070	0.090	0.121	0.110	0.000	0.000	0.065
f1-category/sentiment	0.153	0.204	0.256	0.110	0.159	0.000	0.147
summed f1-score	1.346	1.462	1.821	1.318	0.459	0.873	1.213
ref triplets	128	1018	134	925	67	300	
pred triplets	183	1302	164	1190	136	474	
triplet count difference ratio	1.430	1.279	1.224	1.286	2.030	1.580	1.471
Soft Evaluation							
number of tests	74	446	74	446	60	86	
number of successful tests	57	329	60	316	41	69	
number of error	17	117	14	130	19	17	
success rate	0.770	0.738	0.811	0.709	0.683	0.802	0.752
triplets inner plausibility	73.482	74.470	71.458	74.102	70.875	79.706	74.015
scope	71.125	69.027	68.237	66.781	71.450	80.324	71.157
triplets context aware plausibility	77.732	77.409	76.627	76.283	78.500	84.324	78.479
average rating	73.821	73.302	71.797	72.048	73.325	81.147	74.240

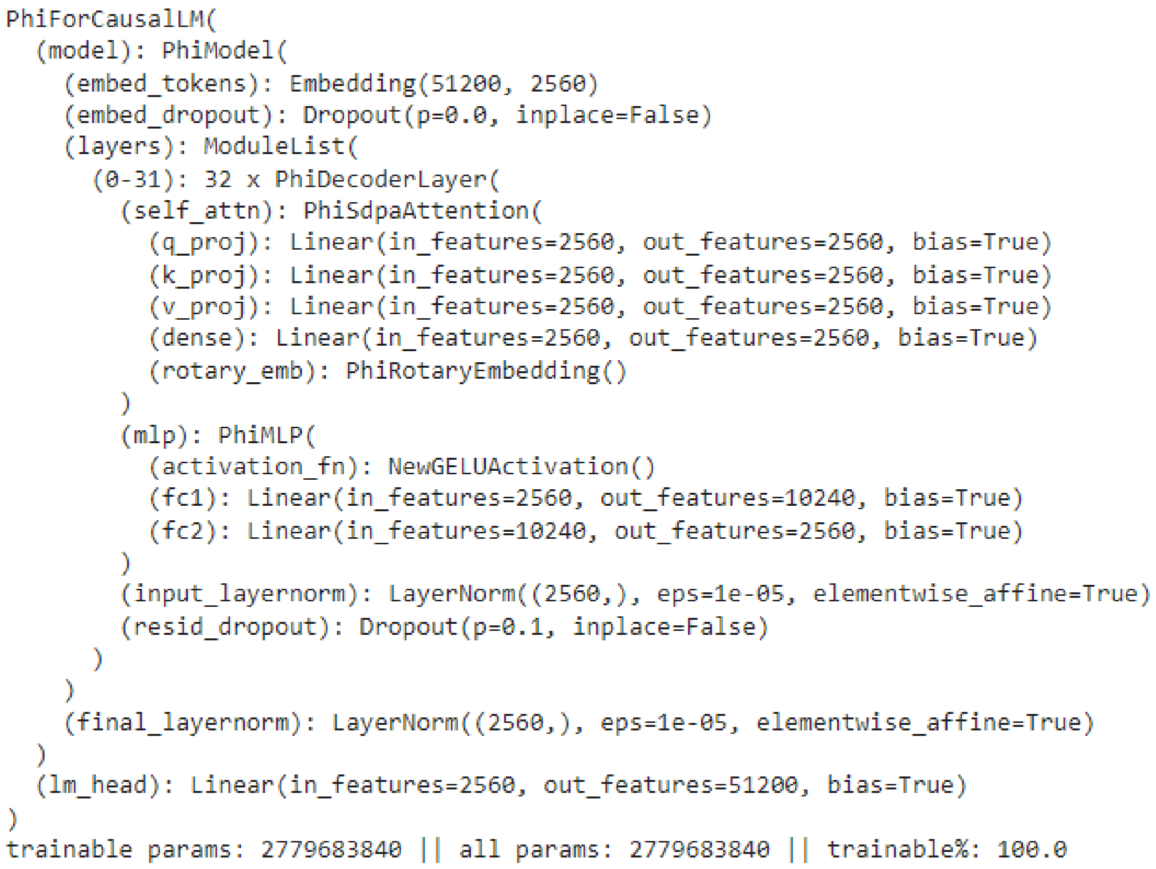
Table C.1 Hard and Soft Evaluation Results for phi-2 Model
Format:
model_name trained_on_dataset tested_on_dataset

Hard Evaluation	llama 3 WA WA	llama 3 WA ALL	llama 3 ALL WA	llama 3 ALL ALL	llama 3 ALL SELF	llama 3 ALL SEMEVAL	average
number of tests	74	446	74	446	60	86	
number of successful tests	45	260	41	358	50	83	
number of error	29	186	33	88	10	3	
success rate	0.608	0.583	0.554	0.803	0.833	0.965	0.724
f1-triplet	0.036	0.054	0.041	0.049	0.000	0.000	0.030
f1-aspect	0.143	0.180	0.151	0.190	0.025	0.002	0.115
f1-category	0.321	0.426	0.628	0.190	0.265	0.000	0.305
f1-sentiment	0.711	0.669	1.000	0.630	0.265	0.002	0.511
f1-category/aspect	0.046	0.076	0.064	0.094	0.000	0.000	0.047
f1-aspect/sentiment	0.103	0.112	0.114	0.126	0.512	0.904	0.312
f1-category/sentiment	0.196	0.112	0.571	0.247	0.012	0.000	0.190
summed f1-score	1.556	1.629	2.360	1.527	1.079	0.909	1.510
ref triplets	116	896	105	1200	86	365	
pred triplets	272	2350	328	1561	239	507	
triplet count difference ratio	2.345	2.623	3.124	1.301	2.779	1.389	2.260
Soft Evaluation							
number of tests	74	446	74	446	60	86	
number of successful tests	48	302	63	400	56	82	
number of error	26	144	11	46	4	4	
success rate	0.649	0.677	0.851	0.897	0.933	0.953	0.827
triplets inner plausibility	72.553	72.421	70.097	73.760	76.455	79.988	74.210
scope	65.723	70.248	52.242	69.870	69.000	78.679	67.630
triplets context aware plausibility	78.489	77.818	64.387	77.325	77.055	83.741	76.470
average rating	71.936	73.162	61.919	73.305	73.873	80.481	72.446

Table C.2 Hard and Soft Evaluation Results for llama-3 Model
Format:
model_name trained_on_dataset tested_on_dataset

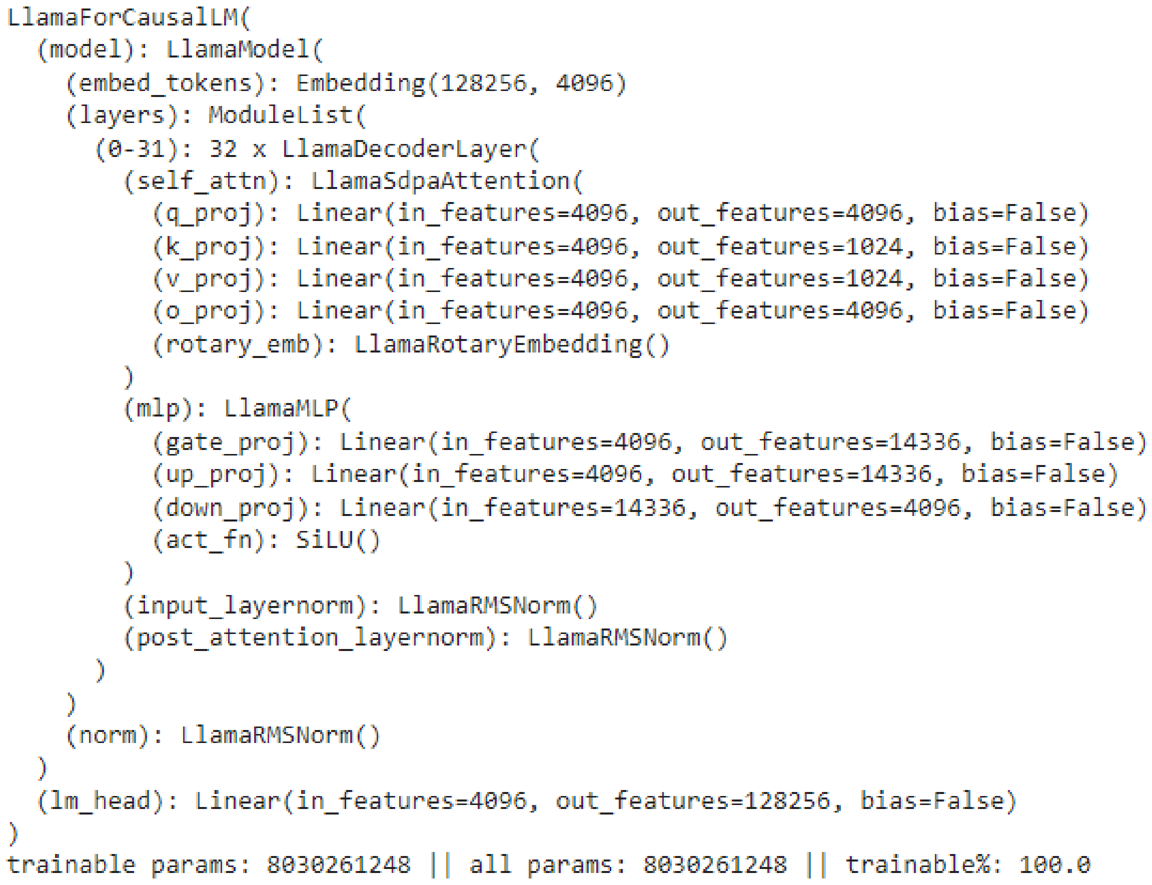
Appendix D

Model Architectures



```
PhiForCausalLM(
  (model): PhiModel(
    (embed_tokens): Embedding(51200, 2560)
    (embed_dropout): Dropout(p=0.0, inplace=False)
    (layers): ModuleList(
      (0-31): 32 x PhiDecoderLayer(
        (self_attn): PhiSdpaAttention(
          (q_proj): Linear(in_features=2560, out_features=2560, bias=True)
          (k_proj): Linear(in_features=2560, out_features=2560, bias=True)
          (v_proj): Linear(in_features=2560, out_features=2560, bias=True)
          (dense): Linear(in_features=2560, out_features=2560, bias=True)
          (rotary_emb): PhiRotaryEmbedding()
        )
        (mlp): PhiMLP(
          (activation_fn): NewGELUActivation()
          (fc1): Linear(in_features=2560, out_features=10240, bias=True)
          (fc2): Linear(in_features=10240, out_features=2560, bias=True)
        )
        (input_layernorm): LayerNorm((2560,), eps=1e-05, elementwise_affine=True)
        (resid_dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (final_layernorm): LayerNorm((2560,), eps=1e-05, elementwise_affine=True)
  )
  (lm_head): Linear(in_features=2560, out_features=51200, bias=True)
)
trainable params: 2779683840 || all params: 2779683840 || trainable%: 100.0
```

Fig. D.1 Phi-2 model architecture



```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(128256, 4096)
    (layers): ModuleList(
      (0-31): 32 x LlamaDecoderLayer(
        (self_attn): LlamaSdpaAttention(
          (q_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (v_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (o_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (rotary_emb): LlamaRotaryEmbedding()
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear(in_features=4096, out_features=14336, bias=False)
          (up_proj): Linear(in_features=4096, out_features=14336, bias=False)
          (down_proj): Linear(in_features=14336, out_features=4096, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm()
        (post_attention_layernorm): LlamaRMSNorm()
      )
    )
    (norm): LlamaRMSNorm()
  )
  (lm_head): Linear(in_features=4096, out_features=128256, bias=False)
)
trainable params: 8030261248 || all params: 8030261248 || trainable%: 100.0
```

Fig. D.2 Llama-3 model architecture

Appendix E

Dataset Pre-Processing

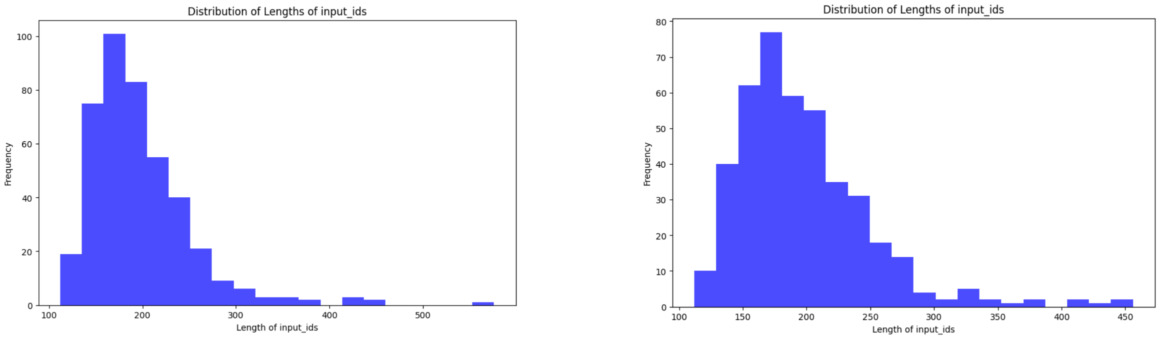


Fig. E.1 Input length of "WhatsApp" dataset before filtering (left) and after (right)

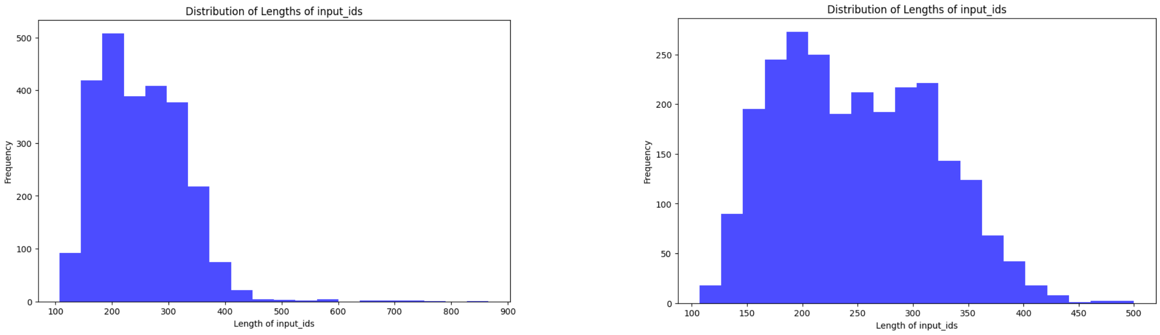


Fig. E.2 Input length of all apps dataset before filtering (left) and after (right)

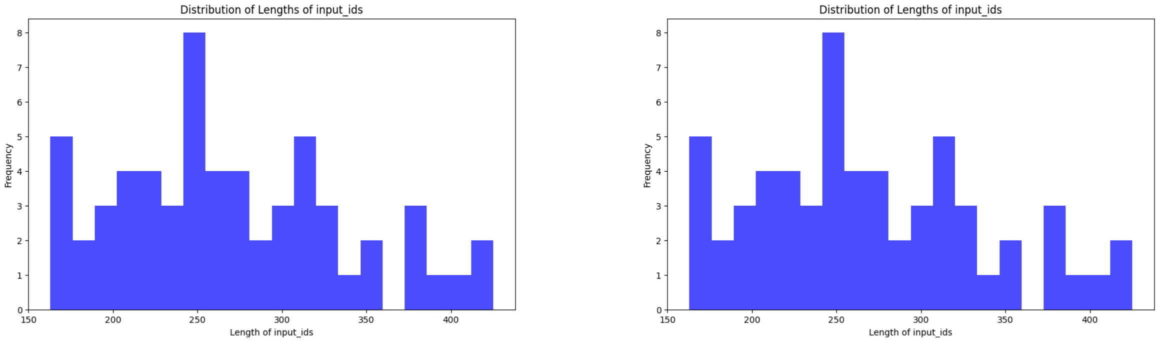


Fig. E.3 Input length of SELF annotated apps dataset before filtering (left) and after (right)

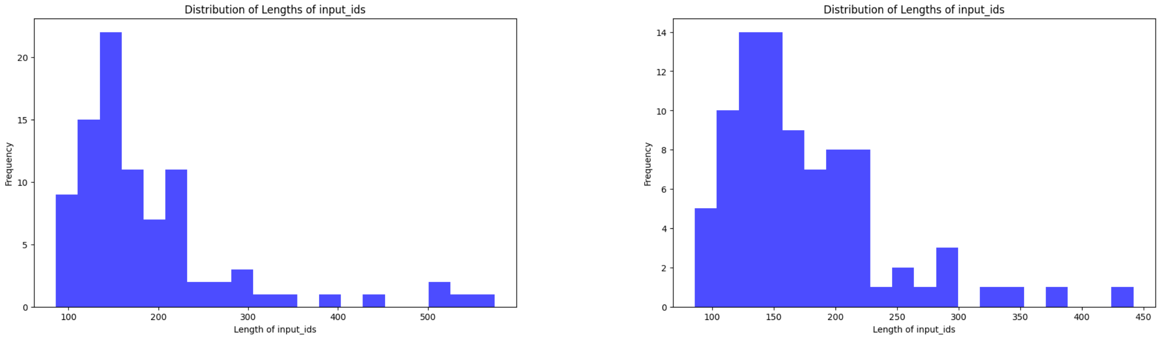


Fig. E.4 Input length of all SemEVal Restaurant dataset before filtering (left) and after (right)