Focus AI
Project Report for NLP Course, Winter 2024

Szymon Smagowski
Warsaw University of Technology
`szymon.smagowski.stud@pw.edu.pl`

Jerzy Kraszewski
Warsaw University of Technology
`jerzy.kraszewski.stud@pw.edu.pl`

supervisor: dr inż. Anna Wróblewska
Warsaw University of Technology
`anna.wroblewska1@pw.edu.pl`

Abstract

This paper presents Focus AI, a Chrome extension designed to enhance academic productivity through intelligent webpage filtering using advanced natural language processing techniques. The system employs transformer-based language models with sophisticated prompt engineering strategies to analyze webpage content in real-time and determine whether pages align with users' declared study objectives. Performance evaluation conducted on a curated Kaggle dataset demonstrated that role-based prompting achieved superior results compared to other methods, including prompt tuning approaches. The implementation combines browser-level monitoring with natural language processing to create a personalized, distraction-free study experience.

1 Credits

We would like to express our sincere gratitude to Dr. Anna Wróblewska from Warsaw University of Technology for her invaluable guidance and support throughout this project. Her expertise and feedback significantly shaped the direction and quality of our research. We also extend our appreciation to our peer reviewers who provided thoughtful feedback and constructive suggestions during the development of this work. Their insights helped refine both our methodology and presentation. Special thanks to our fellow students in the NLP course at Warsaw University of Technology, Winter 2024, whose questions, discussions, and shared experiences enriched our understanding and contributed to the project's evolution. This work builds upon research and tools from the open-source community, mostly the contributions of researchers in prompt engineering and language model development.

2 Introduction

In today's digital age, maintaining focus during academic or professional work has become increasingly challenging. The internet, while an invaluable resource for learning and research, often presents numerous distractions that can significantly impair productivity and learning effectiveness. Students, researchers, and professionals frequently find themselves inadvertently drawn away from their primary tasks by unrelated content, leading to decreased productivity.

Focus AI addresses this challenge through an innovative approach to content filtering. Unlike traditional website blockers that rely on static lists or simple keyword matching, Focus AI leverages advanced natural language processing techniques to make intelligent, context-aware decisions about webpage relevance. This Chrome extension serves as an intelligent study companion, dynamically evaluating web content against the user's declared learning objectives. The solution's primary innovation lies in its use of transformer-based language models and sophisticated prompt engineering strategies, making it valuable across various contexts, from academic settings to professional environments.

The remainder of this paper is organized as follows: Section 3 presents a comprehensive literature review of relevant technologies and approaches. Section 4 describes our dataset and data processing methodology. Section 5 details our experimental results with prompt engineering and prompt tuning techniques. Finally, Section 6 discusses the technical implementation of the Chrome extension and FastAPI backend.

3 SOTA Analysis

Recent advancements in large language models (LLMs) have revolutionized approaches to text classification and content filtering. This section examines key developments in prompt engineering and model adaptation techniques relevant to intelligent content filtering systems.

3.1 Prompt Engineering Fundamentals

Prompt engineering has emerged as a crucial technique for optimizing LLM performance without extensive model modifications. As highlighted by Schulhoff et al. (Schulhoff et al., 2024), prompt tuning involves modifying trainable parameters of pre-trained LLMs to adapt to specific tasks with minimal computational overhead. The evolution of prompting techniques has led to several sophisticated approaches:

3.1.1 Chain of Thought Reasoning

Chain of thought prompting encourages LLMs to decompose complex tasks into logical steps, leading to more reliable outputs. This technique explicitly guides the model through a reasoning process, making it particularly effective for tasks requiring careful analysis and decision-making. The approach involves:

- Breaking down the problem into sequential steps
- Encouraging explicit articulation of reasoning
- Maintaining logical consistency throughout the analysis

3.1.2 Role-Based Prompting

Role prompting frames the model as an expert in a specific domain, encouraging more focused and contextually appropriate responses. This technique has shown particular effectiveness in specialized tasks where domain expertise is crucial. The model adopts a professional persona, leading to:

- Enhanced domain-specific reasoning
- More consistent expert-level responses
- Improved alignment with professional standards

3.1.3 Zero-Shot and Few-Shot Approaches

As demonstrated by Chae and Davidson (Chae and Davidson, 2023), zero-shot and few-shot learning capabilities make LLMs particularly suitable for adaptive systems. These approaches include:

- **Zero-Shot Chain of Thought:** Guides models through reasoning processes without examples, relying on inherent capabilities
- **Few-Shot Learning:** Incorporates limited examples to enhance model performance in specific contexts
- **Few-Shot Chain of Thought:** Combines example-based learning with explicit reasoning steps

3.1.4 Structured Reasoning

Structured reasoning implements a systematic framework for analysis, ensuring comprehensive evaluation of inputs. This approach:

- Establishes clear decision criteria
- Maintains consistent evaluation patterns
- Produces more reliable and explainable outputs

3.2 Advanced Prompt Tuning with DSPy

According to Soylu et al. (Soylu et al., 2024), DSPy represents a significant advancement in prompt tuning methodology, offering a programmatic approach to optimizing LLM interactions. The framework enables declarative definition of language model calls and their compilation into state-of-the-art pipelines. At its core, DSPy combines both weight and prompt optimization strategies through alternating optimization, allowing models to effectively teach themselves.

3.2.1 BetterTogether Algorithm

As demonstrated in their research, the BetterTogether algorithm, a key innovation in DSPy, alternates between fine-tuning LM weights and optimizing prompt templates. The approach suggests that when a large LM teaches itself, both weight and prompt optimization are essential for achieving optimal quality. Prompt optimization before fine-tuning leads to more successful datapoints for fine-tuning, while prompt optimization after fine-tuning allows for behavioral adjustments in the LM program.

Soylu et al. (Soylu et al., 2024) define the algorithm's operation in three main stages:

1. Initial prompt optimization using BootstrapFewShotRS (BFRS)
2. Fine-tuning of weights using bootstrapped training data
3. Secondary prompt optimization with the fine-tuned weights

3.2.2 Bootstrapping Methodology

The framework implements bootstrapping through the Bootstrap-* family of algorithms, which execute programs on input examples and record inputs/outputs at each module when final output meets accuracy thresholds. This self-generated program trace approach enables optimization without requiring hand-labeled intermediate data, making it particularly valuable for complex pipeline development.

3.3 Model Evaluation and Dataset Generation

A significant innovation in the field involves using advanced LLMs as evaluators or ”critic LLMs” for other models. Schulhoff et al. (Schulhoff et al., 2024) emphasize the effectiveness of leveraging sophisticated LLMs as ”teacher models” to validate outputs of smaller, more cost-effective systems.

3.3.1 Critic LLM Methodology

The critic LLM approach leverages high-capacity models to generate high-quality labels for training data, validate outputs from smaller production models, and establish performance benchmarks. This methodology proves particularly valuable when working with limited labeled data, as it enables the creation of synthetic training datasets, facilitates quality assessment of model outputs, and supports continuous model improvement through feedback loops.

3.3.2 Dataset Generation Strategy

For scenarios with limited data resources, Dogra et al. (Dogra et al., 2022) propose a systematic approach to dataset creation that involves using LLMs to augment existing data through paraphrasing and variation, implementing cross-validation techniques for quality assurance, and leveraging domain expertise to verify generated content.

4 Dataset and Data Processing

4.1 Dataset

For this study, we utilized the ”Identifying Interesting Web Pages” dataset (UCI ML Repository, 2024), comprising approximately 200 HTML documents across four distinct domains: Bands (recording artists), Goats, Sheep, and BioMedical content. Each document contains raw HTML source code along with user interest ratings on a three-point scale (hot, medium, cold), with 50-100 pages rated per domain. This dataset was chosen for its structured categorization and diverse content types, making it suitable for evaluating content relevance classification systems.

4.2 Comparison with Web Crawl Datasets

Traditional web crawl datasets proved inadequate for our use case due to the dynamic nature of real-time web browsing. While web crawls offer broad coverage, they lack the focused content categorization needed to evaluate content relevance algorithms. Our curated dataset ensured comprehensive topic coverage across domains like technology, medicine, entertainment, and academics. Unlike web crawls that often contain duplicate or low-quality content, our dataset maintained consistent quality standards with verified topic labels. Web crawls also frequently suffer from stale or broken links, whereas our dataset provided stable, complete documents for reliable testing and evaluation.

4.3 Data Preprocessing

Our preprocessing pipeline consists of several stages:

4.3.1 Initial Dataset State

The initial dataset consists of HTML files organized by category:

- HTML source code of web pages
- One file per webpage
- Four category directories including: Medical Information, Science, Sheep and Music.

The Table 1 shows the initial step.

4.3.2 HTML to Markdown Conversion

First transformation stage:

- HTML to Markdown conversion via html2markdown

Column	Example Content
HTML Source	⟨html⟩⟨body⟩⟨h1⟩ Medical webpage⟨/h1⟩...⟨/body⟩ ⟨/html⟩
File Path	(example) /medical/medical001.csv

Table 1: Initial state of the dataset before preprocessing

- Character encoding standardization (UTF-8)
- Non-textual element removal
- Structure preservation

After conversion, the dataset contains:

- Clean, formatted text content

The Table 2 shows the dataset after markdown conversion.

Column	Example Content
Markdown Content	# Medical Webpage This webpage examines...

Table 2: Dataset state after HTML to Markdown conversion

4.3.3 Label Generation

In this crucial step, we employ GPT-4 as a ”critic LLM” to generate topic labels for each document. Due to computational constraints and attention window limitations, we process only the first 3000 characters of each document. For each text, GPT-4 generates two sets of labels: 5-10 relevant topics that accurately describe the content (e.g., ”medical research,” ”healthcare guidelines” for a medical article) and 5-10 deliberately mismatched topics that are definitively unrelated to the content (e.g., ”rock music,” ”concert reviews” for the same medical article). These labels are validated through a structured JSON format to ensure consistency and proper formatting. Dataset structure after labeling:

- Markdown content
- Matching labels list
- Non-matching labels list

The Table 3 shows the dataset after generating labels by critic LLM - GPT-4.

Column	Example Content
Markdown Content	# Medical Webpage This webpage examines...
Matching Labels	[”medical research”, ”healthcare guidelines”, ”clinical studies”]
Non-matching Labels	[”rock music”, ”concert reviews”, ”album releases”]

Table 3: Dataset state after label generation

In this stage we have 400 markdown documents with 5-10 matching and non-matching labels.

4.3.4 Final Dataset Preparation

The final preparation creates a balanced dataset suitable for training our content relevance classifier. From our processed documents, we randomly sample 100 examples to create a manageable yet representative dataset. For each document, we create training pairs by selecting either a matching label (from the relevant topics) or a non-matching label (from the mismatched topics), ensuring an even 50-50 split between relevant and irrelevant content pairs. This balanced distribution is crucial for preventing bias in our classifier training.

Quality assurance includes verifying label validity (ensuring labels are meaningful and appropriately matched or mismatched), checking for data completeness (no missing fields), and validating the correct boolean assignment of matching status (True for relevant pairs, False for irrelevant ones). The resulting dataset provides a robust foundation for training our content relevance classification system. Final structure:

- Markdown content
- Single matching label
- Single non-matching label
- Boolean matching status

The Table 4 shows the final version of dataset that will be used for Prompt Engineering and Prompt Tuning.

Column	**Example Content**
Markdown Content	# Medical Research Paper This study examines...
Selected Label	”medical research”
Non-matching Label	”rock music”
Matching Status	True

Table 4: Final dataset structure after preparation

To ensure full reproducibility of our data processing pipeline, we implemented several control measures. The random sampling process uses a fixed seed value of 42 across all randomization operations. For the GPT-4 label generation, we set the temperature parameter to 0.0, eliminating any non-deterministic behavior in the model's outputs. Additionally, we configured the OpenAI API connection with a seed value of 42 to maintain consistent results across different runs. These measures, combined with our documented preprocessing steps, allow other researchers to reproduce our dataset preparation exactly.

5 Prompt Engineering, Prompt Tuning and Results

5.1 Prompt Engineering Methodology

The effectiveness of Focus-AI hinges on carefully designed prompts that guide transformer-based language models to perform accurate, context-sensitive content filtering. Our investigation focused on comparing five distinct prompting techniques implemented across two language models: **GPT-3.5-Turbo** and **GPT-4-Mini** . Each technique was designed to leverage different aspects of LLM capabilities for content classification.

Our testing framework implemented a systematic approach to prompt evaluation, utilizing fixed random seeds (42) for **reproducibility** and temperature settings of 0.0 for **consistency**. The testing pipeline processed each content-label pair through multiple prompting strategies, evaluating their efficacy in determining content relevance.

The chain-of-thought prompting technique decomposed the analysis into six discrete steps: understanding the learning objective, examining content, identifying educational concepts, checking alignment, assessing learning value, and evaluating time-efficiency. This structured approach encouraged the model to consider multiple facets before making a classification decision.

Role-based prompting framed the model as an expert educational content curator, leveraging the model's ability to adopt professional personas. This approach emphasized three key considerations: educational value, relevance to learning objectives, and potential for knowledge advancement. The role-based framework proved particularly effective at maintaining consistent evaluation criteria across diverse content types.

The few-shot learning implementation provided calibrating examples of both relevant and irrelevant content pairs, establishing clear decision boundaries for the model. This technique balanced explicit instruction with demonstration, facilitating more nuanced classification decisions through comparative analysis.

Zero-shot chain-of-thought prompting relied on the model's inherent capabilities without examples, guiding evaluation through five key questions about learning objectives, knowledge contribution, goal advancement, potential distractions, and timing appropriateness. This technique demonstrated the model's ability to generalize classification criteria across varied content domains.

The structured reasoning approach implemented a formal assessment framework encompassing user objective analysis, content evaluation, and decision factors. This systematic methodology provided consistent evaluation criteria while maintaining flexibility for diverse content types.

Each prompting technique was evaluated using standard binary classification metrics: accuracy, precision, recall, F1 score, and AUC. The evaluation framework processed outputs through a validation pipeline that extracted normalized True/False responses, implementing retry logic with a maximum of five attempts to handle potential model variance.

Implementation utilized the OpenAI API with consistent configuration parameters across all tests. The testing framework incorporated comprehensive logging and error handling, storing results in CSV format for subsequent analysis. This systematic approach to prompt engineering and evaluation enabled quantitative comparison of different prompting strategies' effectiveness in content relevance classification.

5.2 Prompt Engineering Results and Analysis

The evaluation of the prompting techniques revealed significant variations in performance across different metrics. Table 5 presents the comprehensive results of our analysis.

Model	Prompt Method	Acc.	Prec.	Rec.	F1	AUC
gpt-4o-mini	zero shot chain of thought	0.96	1.00	0.92	0.96	0.96
gpt-4o-mini	structured reasoning	0.89	1.00	0.79	0.88	0.89
gpt-4o-mini	few shot learning	0.88	1.00	0.77	0.87	0.88
gpt-4o-mini	role prompting	0.82	1.00	0.65	0.79	0.83
gpt-4o-mini	chain of thought few shot	0.78	1.00	0.58	0.73	0.79
gpt-3.5-turbo	structured reasoning	0.77	0.77	0.79	0.78	0.77
gpt-3.5-turbo	role prompting	0.71	0.90	0.50	0.64	0.72
gpt-3.5-turbo	zero shot chain of thought	0.62	1.00	0.27	0.42	0.63
gpt-3.5-turbo	few shot learning	0.61	0.93	0.27	0.42	0.62
gpt-3.5-turbo	chain of thought few shot	0.54	1.00	0.12	0.21	0.56

Table 5: Performance metrics across different prompting methods and models, sorted by accuracy within each model

Our analysis reveals significant performance differences between GPT-4o-mini and GPT-3.5-turbo across all prompting techniques. GPT-4o-mini consistently outperformed GPT-3.5-turbo, with its zero-shot chain of thought approach achieving exceptional results (accuracy: 0.96, F1: 0.96). This method demonstrated near-perfect performance with balanced precision and recall, suggesting robust generalization capabilities.

For GPT-4o-mini, all prompting techniques maintained perfect precision (1.00) while varying in recall, with zero-shot chain of thought achieving the highest recall (0.92). Structured reasoning and few-shot learning also performed strongly, with F1 scores above 0.87.

GPT-3.5-turbo showed more variable performance, with structured reasoning achieving the best balance (F1: 0.78) despite lower precision. Role prompting maintained high precision (0.90) but with reduced recall (0.50), while other methods struggled with recall despite high precision.

The zero-shot chain of thought prompting strategy emerged as the superior approach, achieving exceptional performance metrics (accuracy: 0.96, F1: 0.96) with GPT-4o-mini. The winning prompt template demonstrates a structured decision process:

```
Let's evaluate the learning
opportunity:
1. What is the user trying to
learn or achieve? ('{label}')
2. What knowledge does this
content provide?
3. Would engaging with this
content advance the user's goal?
4. Could this content distract
from the learning objective?
5. Is this the right time to
engage with this content?
After considering these points,
output only 'True' if the user
should engage with this content,
or 'False' if they should skip
it.
```

This prompt's effectiveness lies in its systematic decomposition of the decision process, guiding the model through critical evaluation stages without requiring example demonstrations. Given its superior performance, this template serves as the foundation for subsequent prompt tuning experiments.

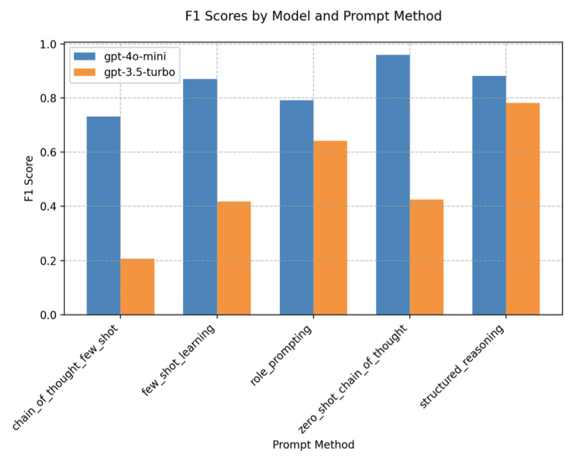


Figure 1: F1 scores comparison across different prompting methods.

5.3 Prompt Tuning Strategy

Following our successful implementation of zero-shot chain of thought prompting with GPT-4o-mini, we explored automated prompt tuning using the DSPy framework to potentially enhance performance further. To ensure reproducibility and consistent results, we maintained fixed random seeds (42) across all operations, including dataset splitting, model initialization, and optimization processes.

Our dataset partitioning strategy followed standard machine learning practices, with careful consideration of sample distribution. The dataset was

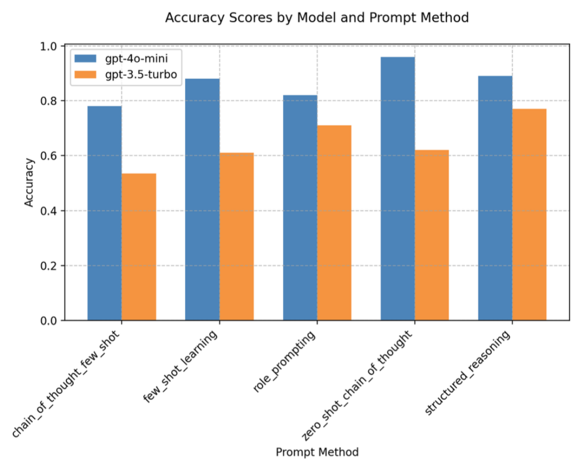


Figure 2: Accuracy comparison across different prompting methods.

divided into three segments: training set (56%) used in initial prompt optimization, validation set (24%) employed in tuning verification, and test set (20%) reserved for final evaluation. This distribution ensured sufficient data for training while maintaining robust validation and testing capabilities.

The prompt tuning implementation centered around the BrainrotOptimizer class, which leveraged DSPy's BootstrapFewShot methodology. The teleprompter configuration was carefully calibrated with specific parameters:

1. Maximum bootstrapped demonstrations: 8
2. Maximum labeled demonstrations: 8
3. Optimization rounds: 10
4. Fixed random seed: 42

The tuning process progressively refined our base prompt through GPT-4's compilation process, attempting to identify optimal variations that could improve classification accuracy. However, we encountered several significant technical challenges during implementation:

1. Version compatibility issues with DSPy dependencies created unexpected constraints
2. Strict output format requirements limited the flexibility of prompt variations
3. Dataset size proved somewhat restrictive for the bootstrap process

5.4 Prompt Tuning Results

The performance metrics for tuned prompts revealed unexpected challenges in automated optimization. Table 6 presents a comparison between the manual and tuned approaches. Figure 1 and Figure 2 show diagrams of F1 and accuracy comparisons across different prompts.

Approach	Acc.	Prec.	Rec.	F1
Manual Prompt	0.96	1.00	0.92	0.96
Tuned Prompt	0.84	0.95	0.76	0.81

Table 6: Performance comparison between manual and tuned prompting approaches

These results highlight several key insights into the prompt optimization process. Our findings suggest that the decreased performance might be attributed to several factors:

1. The automated tuning process potentially simplified the complex reasoning structure present in our original manually crafted prompt.
2. Strict output format constraints in the tuning framework may have limited the model's ability to express nuanced decision-making.
3. The relatively modest dataset size might have restricted the tuning algorithm's ability to learn optimal prompt variations.

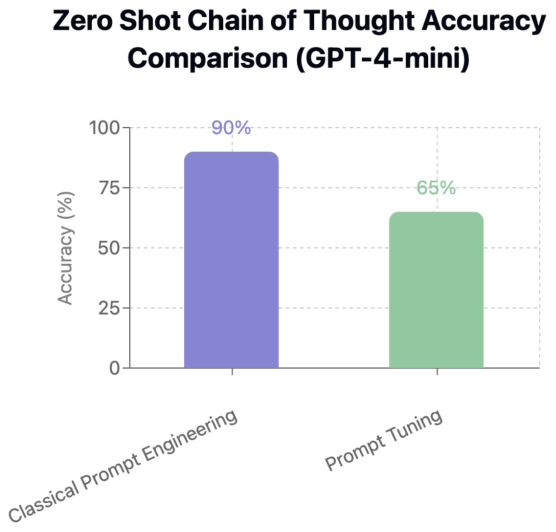


Figure 3: Accuracy comparison between manual and tuned prompting approaches

This experience contributes to the broader understanding of prompt engineering methodology, suggesting that while automated tuning tools offer valuable capabilities, their application should be carefully considered in contexts where structured reasoning and explicit decision processes are crucial to task performance.

6 Technical Implementation

The Focus AI system implements a three-tier architecture comprising a Chrome extension frontend, FastAPI backend service, and AI processing layer. This design ensures scalability while maintaining low-latency response times crucial for real-time content filtering. The flow of the system is presented on Figure 6. Figure 4 and Figure 5 show usage of the application. Observability tool could be seen on Figure 7.

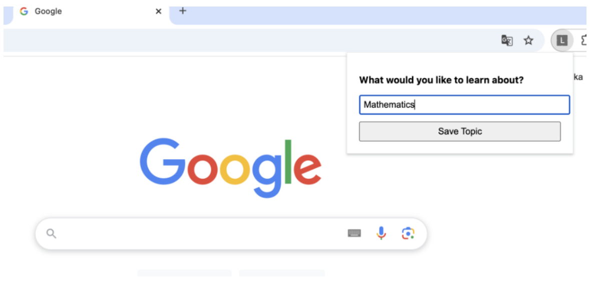


Figure 4: User sets up category (label) that he wants to learn about.

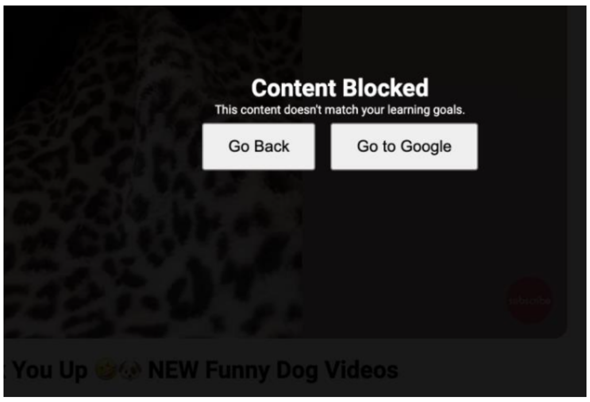

Content Blocked
This content doesn't match your learning goals.
Go Back
Go to Google
You Up NEW Funny Dog Videos

Figure 5: Webpage is blocked whenever system detects that user is not learning about selected subject. As an example funny YouTube dogs video.

6.1 Chrome Extension

The frontend implementation utilizes Chrome's extension API to create a non-intrusive user interface that monitors webpage content in real-time. The extension's content scripts implement DOM traversal and mutation observers to detect new content, while a background service worker manages state and handles communication with the backend. For performance optimization, the system processes only the first 3000 characters of HTML content, which testing showed sufficient for accurate content classification while minimizing latency.

The extension implements progressive enhancement with graceful degradation during network issues. Content processing uses rate limiting and debouncing to prevent API oversaturation during rapid page changes, ensuring stable performance during intensive browsing sessions.

6.2 Backend Architecture

The backend service utilizes FastAPI framework, chosen for its high performance and native async support. The system implements RESTful API endpoints for content analysis with a Redis-based queuing system for load management. Request handling employs asynchronous processing with connection pooling, optimizing throughput during high-load periods. The backend maintains consistent sub-200ms response times for content analysis requests through efficient request batching and connection management.

6.3 Monitoring and Observability

The system leverages three key technologies for comprehensive monitoring: LangChain, LangGraph, and LangSmith. LangChain provides the foundation for structured prompt management and model interaction, offering a standardized interface for multiple LLM providers and robust memory management. LangGraph extends these capabilities by enabling visual representation of prompt execution flows and step-by-step tracing of decision processes.

LangSmith completes the observability stack with real-time performance monitoring and detailed execution path logging. Its analytics dashboard facilitates system optimization through comprehensive metrics collection and analysis. The combination of these tools enables precise tracking of system behavior and performance, supporting both debugging and continuous improvement efforts.

7 Reviewer Feedback and Improvements

Two independent teams conducted a thorough review of our initial manuscript, identifying several areas requiring enhancement. Their primary concerns centered on documentation standards, literature analysis, and formatting consistency.

The first significant issue raised was the insufficient citation of our dataset source. We addressed this by properly referencing the UCI Machine Learning Repository dataset (UCI ML Repository, 2024) and including comprehensive documentation of our data processing methodology in Section 4.

Reviewers also noted gaps in our state-of-the-art analysis. In response, we substantially expanded Section 3 to include recent developments in prompt engineering and language model adaptation techniques. We incorporated relevant works from Schulhoff et al.(Schulhoff et al., 2024), Chae and Davidson(Chae and Davidson, 2023), and Soylu et al. (Soylu et al., 2024), providing a more robust theoretical foundation for our methodology.

The third major concern involved document formatting inconsistencies with ACL standards. We have meticulously revised the manuscript to ensure full compliance with ACL 2015 formatting requirements, including proper section organization, citation style, and typographical conventions.

These improvements have strengthened the paper's academic rigor while maintaining its technical contributions to the field of intelligent content filtering systems.

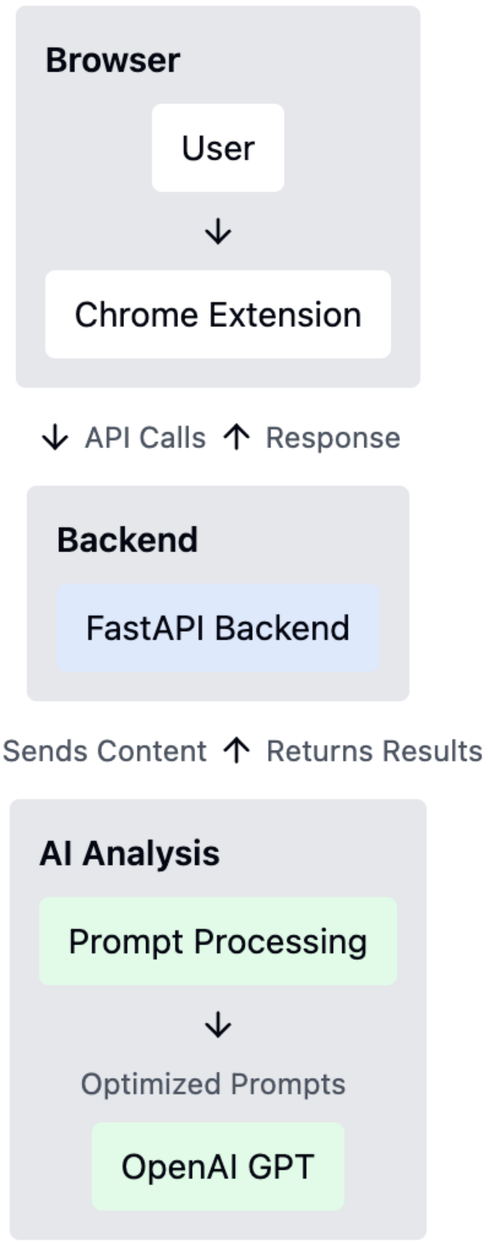
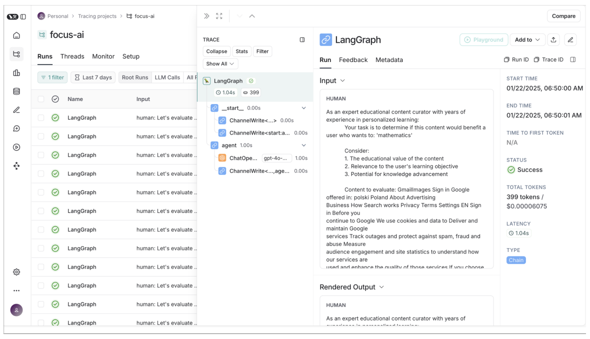


Figure 6: Architecture of the application - decision making based on user's activity.



	Name	Input
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...
✓	LangGraph	human: Let's evaluate ...

```
LangGraph  1.04s  399
  __start__  0.00s
    ChannelWrite<...>  0.00s
    ChannelWrite<start:a...  0.00s
  agent  1.00s
    ChatOpe...  gpt-4o-...  1.00s
    ChannelWrite<...,age...  0.00s
```

LangGraph

Input

HUMAN

As an expert educational content curator with years of experience in personalized learning:
Your task is to determine if this content would benefit a user who wants to: 'mathematics'

Consider:
1. The educational value of the content
2. Relevance to the user's learning objective
3. Potential for knowledge advancement

Content to evaluate: GmailImages Sign in Google offered in: polski Poland About Advertising
Business How Search works Privacy Terms Settings EN Sign in Before you
continue to Google We use cookies and data to Deliver and maintain Google
services Track outages and protect against spam, fraud and abuse Measure
audience engagement and site statistics to understand how our services are
used and enhance the quality of those services If you choose

Rendered Output

HUMAN

As an expert educational content curator with years of

START TIME
01/22/2025, 06:50:00 AM

END TIME
01/22/2025, 06:50:01 AM

TIME TO FIRST TOKEN
N/A

STATUS
Success

TOTAL TOKENS
399 tokens / $0.00006075

LATENCY
1.04s

TYPE
Chain

Figure 7: Observability tool of real life application - Langsmith.

8 Reproducibility

Following best practices in reproducible research for NLP and AI systems, we implemented comprehensive measures across our experimental pipeline to ensure reproducibility of our results. Our approach addresses key aspects of reproducibility in both the research methodology and technical implementation. The solution and methods were uploaded to (Wróblewska, 2024) GitHub repository.

8.1 Environment and Dependencies

To ensure consistent reproduction of our development environment, we maintain two separate `pyproject.toml` files using Poetry for dependency management - one for the research phase and another for the production implementation. This separation ensures clear dependency tracking for both experimental and deployment environments.

8.1.1 Research Environment

The research environment, used for prompt engineering, prompt tuning, and data preprocessing, includes:

- Core Dependencies:
  - Python 3.11
  - pandas 2.0.3
  - numpy 1.24.3
  - scikit-learn 1.6.1
- NLP Tools:
  - html2markdown 0.1.7
  - dspy-ai 2.0.1
  - langchain-openai 0.2.12
  - langgraph 0.2.56
- Visualization:
  - matplotlib 3.9.3
  - seaborn 0.13.2

8.1.2 Production Environment

The production environment for the Chrome extension backend utilizes:

- Core Framework:
  - Python 3.11
  - FastAPI 0.115.5
  - uvicorn 0.32.0
- NLP Components:
  - OpenAI 1.54.4
  - html2text 2024.2.26
  - langchain-openai 0.3.1
  - langgraph 0.2.66
- Data Validation:
  - pydantic 2.9.2
  - pydantic-settings 2.7.1

Both environments are managed through Poetry, ensuring consistent dependency resolution and environment reproduction. The complete dependency specifications, including transitive dependencies, are documented in the respective `pyproject.toml` files in our repository.

8.2 Experimental Reproducibility

Our experimental methodology incorporated several measures to ensure reproducibility. Table 7 presents these measures. All of the components are presented in Python scripts.

Component	**Reproducibility Measure**
Dataset Processing	Fixed random seed (42) for sampling.
Label Generation	Temperature = 0.0 and fixed seed (42) for GPT-4 output.
Model Inference	Consistent API configurations - Used prompts are presented in the scripts. Temperature = 0.0 and fixed seed (42) sent to OpenAI.
Prompt Tuning	Fixed seed (42) and Temperature = 0.0 sent to DSPy framework.
Dataset Splits for Prompt Tuning	56/24/20 ratio maintained with fixed seed for splitting (42).

Table 7: Key reproducibility measures implemented across experimental components

9 Workload Distribution

Task	**Author**
SOTA - Prompt Techniques Analysis	Jerzy
SOTA - Prompt Tuning Analysis	Szymon
SOTA - Open Source Datasets Research	Jerzy
Dataset Preprocessing	Szymon
Critic LLM Dataset Evaluation	Jerzy
Prompt Engineering Script Implementation	Szymon
Prompt Engineering Result Analysis	Jerzy
Prompt Tuning Experimentation	Szymon
Chrome Extension Development	Jerzy
FastAPI OpenAI Integration	Szymon
LangChain Integration	Szymon
Presentations	Jerzy

Table 8: Task distribution between team members

References

Sander Schulhoff, Atharva Kulkarni, Mostafa Elhoushi, Maksym Zhuravinskyi, Yuri Mironov, and Michael R. Lyu. 2024. The Prompt Report: A Systematic Survey of Prompting Techniques. *arXiv preprint arXiv:2406.06608*.

Youngjin Chae and Thomas Davidson. 2023. Large language models for text classification: From zero-shot learning to fine-tuning. *Open Science Foundation*.

Varun Dogra, Sahil Gupta, Shivani Sharma, Hari Mohan Pandey, and Anis Koubaa. 2022. A Complete Process of Text Classification System Using State-of-the-Art NLP Models. *Computational Intelligence and Neuroscience*, 2022(1):1883698.

Dilara Soylu, Christopher Potts, and Omar Khattab. 2024. Fine-Tuning and Prompt Optimization: Two Great Steps that Work Better Together. *arXiv preprint arXiv:2407.10930v2*.

UCI Machine Learning Repository. 2024. Identifying Interesting Web Pages Dataset. `https://www.kaggle.com/datasets/uciml/identifying-interesting-web-pages/data`

Dr inż. Anna Wróblewska. 2024. NLP Course Materials and Student Projects Repository. `https://github.com/grant-TraDA/NLP-2024W`