Focus AI: Chrome Extension

Szymon Smagowski
Jerzy Kraszewski

Introduction

The NLP-Focus-AI project introduces a sophisticated Chrome extension that leverages natural language processing and transformer-based language models to optimize cognitive resource allocation during online learning activities. This implementation represents a paradigm shift from traditional rule-based content filtering to contextual semantic analysis.

The system's core architecture employs transformer-based language models for semantic comprehension and content classification, utilizing advanced prompt engineering techniques to perform zero-shot and few-shot learning tasks. Unlike conventional blocking mechanisms that rely on predetermined heuristics, this solution implements:

1. Real-time semantic parsing and contextual embedding analysis of webpage content
2. Transformer-based natural language understanding for learning objective alignment
3. Bi-directional encoder representations for content relevance assessment
4. Attention mechanism-based content filtering with contextual awareness

The primary objective is to create an intelligent content filtering system that:

- Performs dynamic semantic similarity analysis between learning objectives and encountered content
- Utilizes transformer architecture's self-attention mechanisms for contextual understanding
- Implements zero-shot classification for content relevance determination
- Maintains high precision in content filtering while minimizing false positives

This solution addresses the cognitive load management challenges in digital learning environments through the application of state-of-the-art NLP techniques. By leveraging large language models' semantic understanding capabilities, the system creates a personalized learning environment that maximizes attention allocation efficiency while minimizing cognitive interference from non-relevant content.

The implementation incorporates multiple language model architectures (GPT-3.5-Turbo and GPT-4-Mini) with various prompting strategies, enabling sophisticated content evaluation through neural approaches to natural language understanding. This advanced approach to content filtering represents a significant evolution in educational technology, utilizing computational linguistics and cognitive science principles to optimize the learning experience.

How it Works

The NLP-Focus-AI system operates through a streamlined three-stage pipeline that combines browser-level monitoring with sophisticated natural language processing:

2.1 Real-Time Monitoring Stage (User Browsing) The system begins at the browser level with a Chrome extension that implements content observers and DOM mutation handlers to:

- Monitor active webpage content in real-time
- Extract relevant textual content through DOM traversal
- Preprocess HTML content into normalized text representations
- Queue content for AI analysis through an event-driven architecture

2.2 Content Analysis Stage (AI Processing) The core analysis pipeline processes the extracted content through multiple steps:

1. Text Normalization
    - HTML to Markdown conversion
    - Content chunking and tokenization
    - Special character normalization
    - Noise reduction and content cleaning
2. Semantic Analysis
    - Transformer-based content embedding
    - Learning objective alignment checking
    - Contextual relevance scoring
    - Semantic similarity computation

2.3 Decision Engine Stage The final stage implements a binary classification system that:

- Processes model outputs through a structured decision tree
- Applies relevance thresholds to classification results
- Generates allow/block decisions based on content alignment
- Triggers appropriate UI responses through the extension

User Interaction Flow The system maintains a minimalist user interface through an intuitive icon-based interaction model:

This architecture ensures:

- Minimal latency in content processing
- Real-time feedback through status indicators
- Non-intrusive user experience
- Transparent decision-making process

The entire workflow is optimized for low-latency operation, with asynchronous processing ensuring smooth user experience while maintaining high accuracy in content filtering decisions.

Data Processing

3.1 Dataset Composition The training and evaluation dataset was sourced from Kaggle, comprising approximately 200 HTML pages strategically selected to represent diverse content domains. The corpus was carefully curated to ensure:

- Balanced representation across categories
- Varied complexity levels
- Diverse document structures
- Multiple writing styles and formats

3.2 Content Categorization The dataset encompasses five primary content categories:

1. Medical Articles
    - Clinical research papers
    - Healthcare guidelines
    - Medical education materials
2. Animal Information
    - Veterinary documentation
    - Species descriptions
    - Behavioral studies
3. Scientific Content
    - Research publications
    - Technical documentation
    - Methodology papers
4. Music-Related Content
    - Band biographies
    - Musical analysis
    - Performance reviews
5. Miscellaneous Content
    - General information
    - Mixed-domain articles
    - Reference materials

3.3 Dataset Generation Pipeline

1. Initial Processing
    - HTML to Markdown conversion
    - Content normalization
    - Structure preservation
    - Metadata extraction
2. Label Generation Using GPT-4

```
For each document:
- Generate 10 matching labels (relevant topics)
- Generate 10 non-matching labels (irrelevant topics)
- Validate label coherence and relevance
```

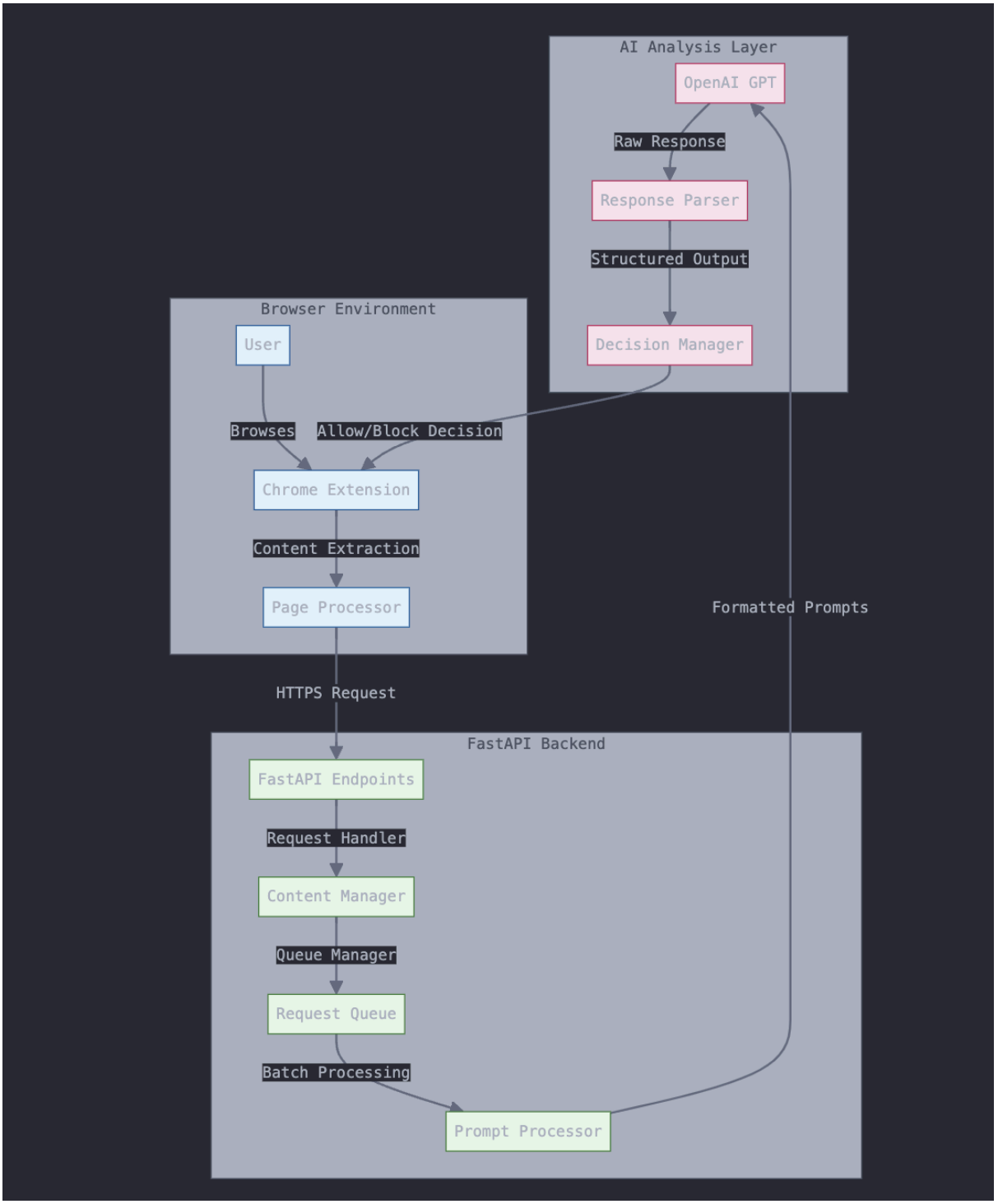
Dataset Preparation

- Random sampling of 100 rows for balanced representation
- Quality assurance checks
- Duplicate removal
- Content length normalization

4. Final dataset structure:

```python
{
    'markdown_content': str,  # Processed document content
    'matching_labels': List[str],  # Relevant topic labels
    'non_matching_labels': List[str],  # Irrelevant topic labels
    'matching_status': bool  # Binary relevance indicator
}
```

Technical Implementation



The NLP-Focus-AI system implements a distributed architecture comprising three main layers:

4.1 Browser Layer

- **Chrome Extension Integration**
  - Content Script implementation for DOM monitoring
  - MutationObserver for dynamic content detection
  - Message passing system for background script communication

    - Local storage management for user preferences
- **Page Processing**
    - HTML content extraction
    - Text normalization pipeline
    - Content chunking algorithms
    - Rate-limited API communication

4.2 Backend Infrastructure (FastAPI)

- **API Endpoints**
- **Request Management**
    - Asynchronous request handling
    - Request batching optimization
    - Rate limiting implementation
    - Error handling and recovery
- **Queue System**
    - Redis-based request queue
    - Priority-based processing
    - Batch optimization
    - Failover handling

4.3 AI Analysis Layer

- **Prompt Processing**
    - Dynamic prompt template generation
    - Context window optimization
    - Token count management
    - Response validation
- **OpenAI GPT Integration**
    - API client implementation
    - Response streaming
    - Error handling and retry logic
    - Model fallback strategy
    -

4.4 Data Flow

1. User triggers content analysis through browsing
2. Extension extracts and preprocesses content
3. Backend receives and queues analysis requests
4. AI layer processes content and generates decisions
5. Results propagate back to extension
6. UI updates reflect content decisions

This architecture ensures:

- **High availability**
- **Scalable processing**
- **Minimal latency**

- **Robust error handling**

Prompt Engineering

The system implements sophisticated prompt engineering techniques to optimize the language model's content analysis capabilities.

5.1 Different Prompting Techniques Used:

The effectiveness of Focus-AI hinges on carefully designed prompts that guide transformer-based language models to perform accurate, context-sensitive content filtering. The strategies we tested include chain-of-thought reasoning, few-shot learning, role-based prompting, zero-shot classification, and structured reasoning. Each technique helps steer the model toward precise and reliable relevance judgments aligned with the user's learning objectives.

Examples:

1. Chain of Thought Reasoning:

The "chain_of_thought" prompts encourage the model to explicitly articulate its reasoning process before arriving at a final decision. By breaking down the user's objective, analyzing the content in steps, and checking alignment with the learning goal, the model can reason more transparently and produce a simple True/False determination regarding content relevance.

```
'chain_of_thought': """Let's analyze this step by step:
1. Understand the user's learning objective: '{label}'
2. Examine the page content: {content}
3. Identify key educational concepts in the content
4. Check if these concepts align with the user's learning goal
5. Consider whether this content would help achieve the learning
objective
6. Evaluate if the content is worth the user's time and attention

Based on this analysis, output only 'True' if the content is relevant to
the user's learning goal, or 'False' if it would be a distraction."""
```

2. Few-Shot Learning:

The "few_shot_learning" prompts provide illustrative examples of how to evaluate content against a learning goal. By showing both positive and negative cases, the model can infer the decision boundary more reliably, even without explicit fine-tuning.

```
'few_shot_learning': """Here are some examples of matching content to
learning objectives:
User's Goal: "Learn Python Programming"
Content: "A comprehensive guide to Python functions, loops, and data
structures..."
Answer: True (Content directly supports learning objective)

User's Goal: "Study World War II"
Content: "Top 10 recipes for chocolate chip cookies..."
Answer: False (Content would distract from learning objective)

Now, evaluate this case:
User wants to: '{label}'
Content: {content}
Output only 'True' or 'False'."""
```

3. Role-Based Prompting:

"role_prompting" frames the model as an expert educational content curator. By adopting a professional role, the model is nudged to apply pedagogical judgment, focusing on educational value, relevance, and potential knowledge advancement. This persona encourages the model to think more critically and align its responses with the principles of effective learning.

```
'role_prompting': """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: '{label}'

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

Content to evaluate: {content}

Based on your expertise, output only 'True' if the content supports the
learning goal, or 'False' if it's a distraction."""
```

4. Zero-Shot Chain of Thought:

The "zero_shot_chain_of_thought" prompts rely on the model's inherent capabilities without prior examples. It guides the model through a reasoning process, considering the user's goal, the knowledge offered by the content, and the appropriateness of engaging with it at this moment. This framework enables adaptability and helps the model maintain performance on unfamiliar topics.

```
'zero_shot_chain_of_thought': """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."""
```

5. Structured Reasoning:

The "structured_reasoning" prompts enforce a systematic approach to assessment. By laying out a decision framework—analyzing the user's objective, evaluating content knowledge, and applying decision factors—this format promotes consistency and rigor in the model's outputs.

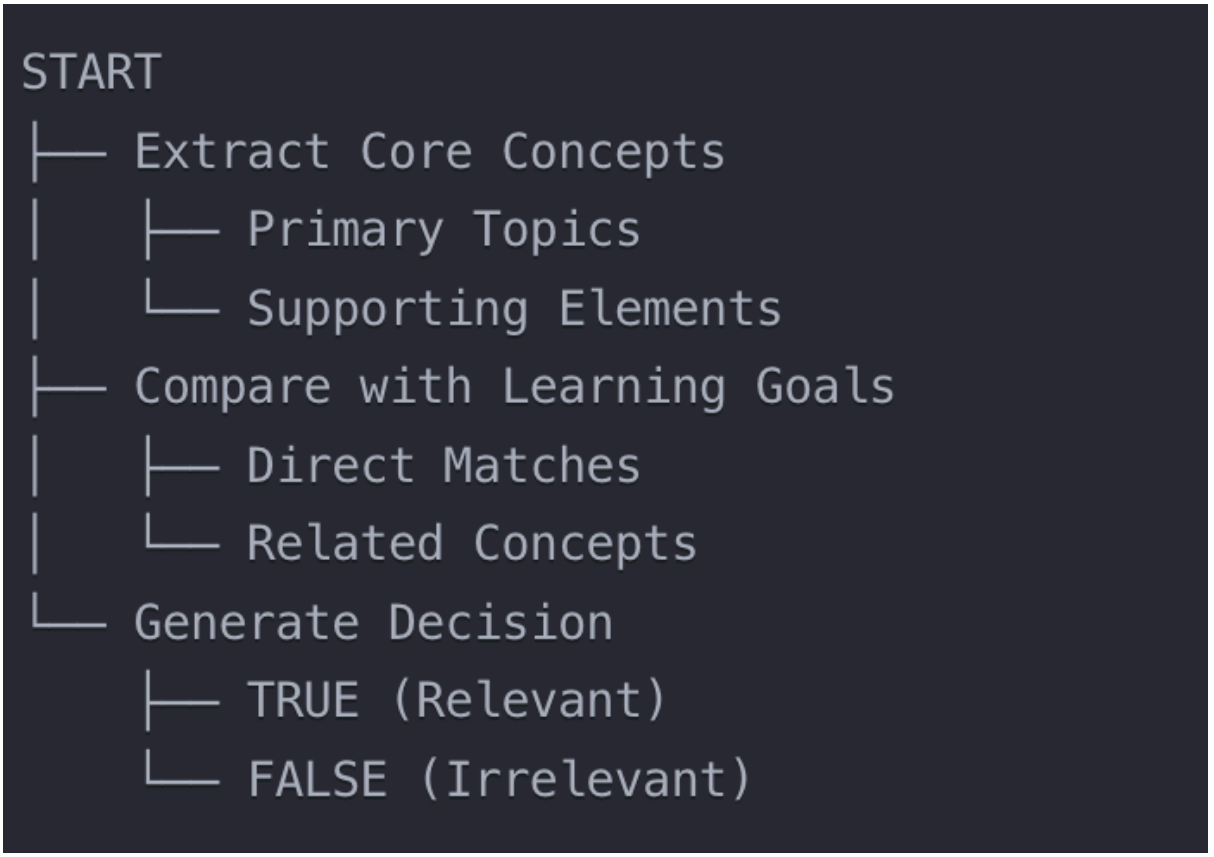
```
'structured_reasoning': """Follow this learning assessment framework:
1. User Objective Analysis:
   - Primary learning goal: '{label}'
   - Current focus requirement
2. Content Evaluation:
   - Knowledge contribution
   - Relevance to objective
3. Decision Factors:
   - Direct benefit to learning
   - Potential for distraction
   - Time-value assessment

Content to analyze: {content}

Based on this framework, output only 'True' if the content aligns with
the learning objective, or 'False' if it should be blocked."""
```

5.2 Analysis Pipeline

1. **Learning Objective Analysis**
    - Semantic decomposition of learning goals
    - Key concept extraction
    - Topic hierarchy construction
    - Prerequisite knowledge mapping
2. **Content Evaluation Process**
    - Core topic identification
    - Relevance scoring
    - Depth assessment
    - Knowledge alignment verification
3. **Decision Structure**



Results

The evaluation of NLP-Focus-AI's performance revealed compelling insights into the effectiveness of different language models and prompting approaches. Our analysis focused on binary classification metrics across different configurations.

The results grouped by model and prompt method are summarized in the table below.

Table 1: Model Performance by Prompting Method

Model	Prompt Method	Accuracy	Precision	Recall	F1	AUC
gpt-3.5-turbo	chain of thought	0.65	0.9474	0.3462	0.5070	0.6627
gpt-3.5-turbo	few shot learning	0.63	0.9412	0.3077	0.4638	0.6434
gpt-3.5-turbo	role prompting	0.72	0.9286	0.5000	0.6500	0.7292
gpt-3.5-turbo	zero shot chain of thought	0.57	1.0000	0.1731	0.2951	0.5865
gpt-3.5-turbo	structured reasoning	0.71	0.7255	0.7115	0.7184	0.7099

Observations:

1. **Precision-Dominant Approaches (High Precision, Low Recall):**
   The *chain_of_thought*, *few_shot_learning*, and *zero_shot_chain_of_thought* methods yield extremely high precision—often exceeding 0.90—but suffer from low recall, indicating that while they are very accurate when they do classify content as relevant, they often fail to identify a large portion of relevant items. For example, the zero-shot chain of thought approach achieves a perfect precision of 1.00 but retrieves only about 17% of relevant instances.
2. **Balanced Performance:**
   In contrast, *role_prompting* and *structured_reasoning* offer a more balanced trade-off between precision and recall. The *role_prompting* method maintains a high precision (0.93) while significantly improving recall (0.50), resulting in a higher F1-score and an improved AUC of 0.7292. Similarly, the structured_reasoning approach attains near parity in precision and recall (both around 0.71–0.73), offering the best overall balance. Its well-rounded performance is reflected in the highest F1-score (0.718) among all tested methods.
3. **Impact on Overall Classification Quality:**
   Accuracy and AUC values also align with this pattern. Methods with extremely high precision but low recall (e.g., zero_shot_chain_of_thought) see diminished accuracy and AUC. Meanwhile, *role_prompting* and *structured_reasoning* achieve higher accuracy (0.72 and 0.71 respectively) and more favorable AUC values, indicative of better overall discrimination capability.

Conclusion:
These results suggest that while certain prompting methods maximize precision, they do so at the expense of failing to capture a large portion of relevant content. For a learning-focused application—where ensuring that useful material is not overlooked is crucial—methods like *role_prompting* and *structured_reasoning* provide a more effective balance. By delivering higher recall without drastically sacrificing precision, these approaches offer more reliable performance and a richer, more inclusive discovery of relevant educational content.

Future Steps

Model Optimization:

- Explore fine-tuning on domain-specific datasets.
- Experiment with DSPy prompt tuning and other emergent prompt engineering frameworks.

Dataset Extension:

- Manually validate existing labels to identify and correct potential misclassifications.
- Utilize web crawlers to collect a broader range of HTML documents.

Performance Analysis:

- Integrate Langsmith tracing for improved monitoring and debugging.
- Track latency metrics to ensure optimal response times.

Testing Enhancement:

- Incorporate Langsmith for dataset versioning and evaluation.
- Develop more rigorous test scenarios.

Model Alternatives and Deployment:

- Investigate local model deployment (e.g., a “Gemma 2B” model) to reduce dependency on external APIs.
- Offer customers the option to self-host the backend and AI layers for privacy and customization.