Streaming, Fast and Slow: Cognitive Load-Aware Streaming for Efficient LLM Serving

Chang Xiao
Boston University
Boston, USA
cxiao1@bu.edu

Brenda Z. Yang
Columbia University
New York, USA
zy2231@columbia.edu

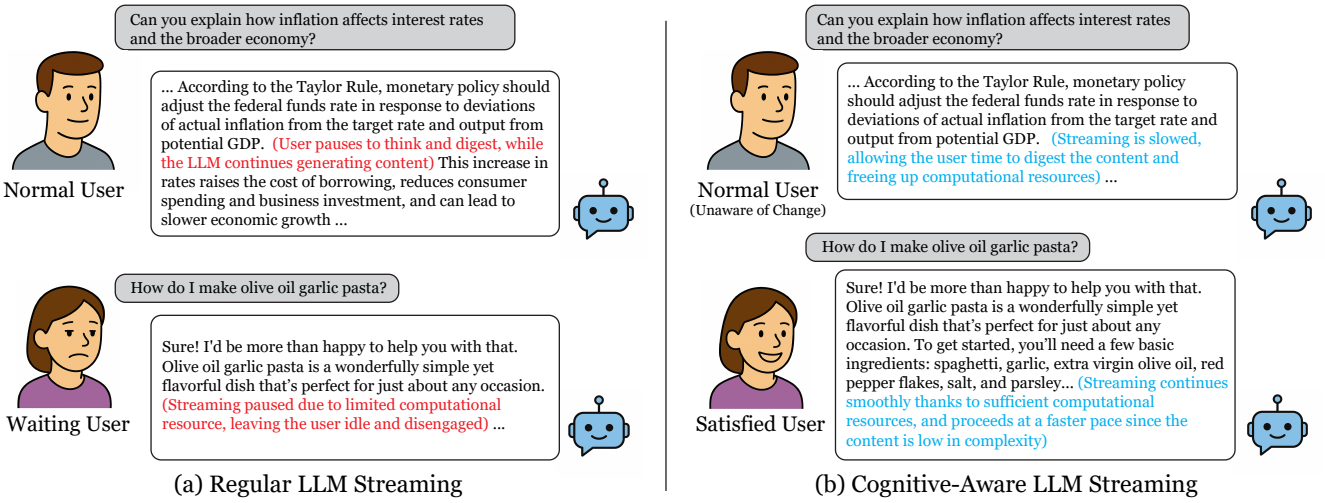

Figure 1: (a) In regular LLM streaming, the streaming speed is not associated with the content being delivered. Complex content may be streamed faster than users can comfortably read, resulting in wasted resource. Conversely, simple content may be streamed too slowly, causing users to wait unnecessarily. (b) In our proposed approach, the streaming speed is adapted based on the estimated content complexity. Complex content is slowed down to support user comprehension and optimize resource usage, while simpler content is streamed more quickly to better align with the user's natural reading speed.

Abstract

Generative conversational interfaces powered by large language models (LLMs) typically stream output token-by-token at a rate determined by computational budget, often neglecting actual human reading speeds and the cognitive load associated with the content. This mismatch frequently leads to inefficient use of computational resources. For example, in cloud-based services, streaming content faster than users can read appears unnecessary, resulting in wasted computational resources and potential delays for other users, particularly during peak usage periods. To address this issue, we propose an adaptive streaming method that dynamically adjusts the pacing of LLM streaming output in real-time based on inferred cognitive load. Our approach estimates the cognitive load associated with streaming content and strategically slows down the stream during complex or information-rich segments, thereby freeing computational resources for other users. We conducted a statistical analysis and simulation based on a statistical model derived from data collected in a crowdsourced user study across various types of LLM-generated content. Our results show that this adaptive method can effectively reduce computational consumption while largely maintaining streaming speed above user's normal reading speed.

CCS Concepts

• **Human-centered computing** → **Natural language interfaces**.

Keywords

Cognitive Load; Large Language Models; Human-AI Interaction; Adaptive Text Streaming

1 Introduction

Conversational AI have rapidly gained popularity and utility since their release. Trained on large-scale corpora using high-capacity deep models, these systems offer highly interactive and informative experiences, allowing users to access knowledge and engage in conversations across diverse domains simply by asking questions in natural language [58].

Most user-facing LLM services, including widely used platforms like ChatGPT, are hosted on cloud infrastructures. While this setup offers broad accessibility, it also poses significant challenges in computational resource management. As of early 2025, ChatGPT has experienced substantial growth in user engagement, serving over 400 million weekly active users who collectively send more than 1 billion messages daily [74]. During peak usage periods, demand frequently exceeds the available computational capacity, causing

user requests to queue, leading to increased waiting times and degraded user experiences [79]. Figure 1(a) illustrates a common scenario: when a user submits a query, the LLM generates output at its maximum available speed. However, rapid generation may be unnecessary when the content is dense or cognitively demanding—for example, complex sentence structures, intricate logical arguments, or uncommon terminology—as users often need to pause or slow down their reading. Despite this, the LLM continues streaming at full pace regardless of content, resulting in wasted computational resources. At the same time, other users may experience prolonged waiting times during peak periods due to insufficient resources. Thus, there is a growing need for adaptive optimization of text streaming in LLMs to enhance both system efficiency and user experience.

Prior to the emergence of LLMs, adaptive text streaming had received little attention in both research and practice. This was largely because traditional text-based content (*e.g.,* articles, emails, or web pages) was typically delivered either in full or at speeds far exceeding human reading rates, allowing users to consume the content at their own pace. As a result, there was minimal concern about computational resource constraints or the need for fine-grained control over text delivery speed.

In contrast, modern chat-based LLMs have brought the challenge of text streaming to the forefront. Because LLMs require substantial computational power and generate content token by token, their responses are typically streamed to users at a rate determined by the available computational resources and model complexity. This means that users must consume the content incrementally, as it is generated in real time. Although in some scenarios users may wait for the full output to be generated before reading (e.g., in code generation), in practice, most chat-based LLM interfaces are designed to support immediate consumption as whenever output tokens becomes available. This streaming interaction model has become the norm because it gives users quicker feedback, a greater sense of responsiveness, and the opportunity to interrupt or re-prompt based on partial output [8, 48, 49, 52]. Crucially, this shift transforms text from a static payload into a dynamic, time-based medium. Combined with the cloud-hosted nature of LLMs and their massive user bases, it raises timely and important questions about how to stream text efficiently and adaptively for better computational resource management.

Traditional adaptive streaming methods employed in multimedia services typically rely on static or rule-based strategies, such as buffer-based throttling [31] or quality adjustment mechanisms [68]. While these techniques could be adapted for LLM-based services, they primarily operate from a system-level perspective and overlook the significant differences in users' cognitive load when reading various LLM-generated content. Variations in complexity and informational density can directly affect the reading speed of users. This situation poses an intriguing research question: **Can we dynamically identify text segments where users experience increased cognitive load and strategically adjust the streaming pace, thereby reallocating computational resources more efficiently?**

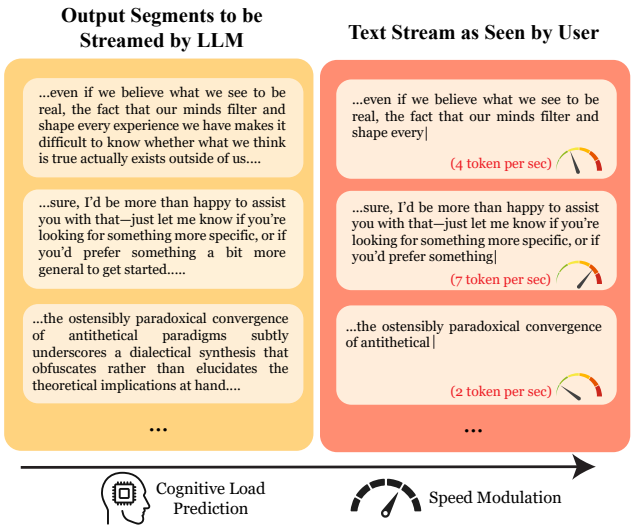

Figure 2: Overview of the workflow of our cognitive-aware streaming framework. Content with high cognitive load is streamed at a lower speed to give users more time to process it, while content with low cognitive load is streamed at a higher speed to minimize unnecessary waiting.

To address this, we propose a novel adaptive streaming framework specifically designed for LLM outputs by incorporating real-time cognitive load assessment to modulate streaming speeds. Figure 2 provides a high-level overview of the system architecture. We conducted a statistical analysis and simulation based on a statistical model derived from data collected in a crowdsourced user study ($N = 300$) across various types of LLM-generated content. The results show that, compared to a standard streaming strategy, our approach can significantly reduce computational resource usage while still delivering content above the user's normal reading speed.

In summary, the contributions of this paper are threefold:

- We introduce the concept of adaptive text streaming based on the cognitive load of the content to be streamed, aimed at improving the efficiency of LLM serving in conversational AI applications.
- We present a statistical analysis framework that enables practitioners to quantify potential computational resource savings in their own scenarios.
- We develop a prototype system that estimates user reading speed and dynamically adjusts the streaming speed. Our experiments demonstrate that our method effectively reduces computational usage while largely ensuring that users receive content above their normal reading speed.

2 Related Work

2.1 Human Reading Speed of Textual Content

Reading speed in digital environments is a topic of active research across HCI, cognitive psychology, education, and information science [23, 42]. Numerous factors — from display technology [73] and screen size [20] to typography [83], layout [19], and user interface

design [61] — have been studied for their impact on how quickly people can read text on screens.

While these factors play an important role in shaping how humans consume text, they typically remain fixed during a user's interaction with a conversational AI system. Instead, what varies is the content itself, as users submit different requests and receive diverse responses. Therefore, the textual content itself—and more specifically, the cognitive load it imposes—becomes the dominant factor influencing how quickly a user can process the information [14, 21]. Consequently, cognitive load can be used as the primary signal for estimating human reading speed in the context of human-LLM interactions.

Cognitive Load Estimation. The Cognitive Load Theory (CLT) [64, 76], one of the most widely accepted frameworks for understanding how humans process information, categorizes cognitive load into three types: intrinsic load, determined by the complexity of the content (e.g., how much information the content contains); extraneous load, related to how the information is presented (e.g., the complexity of sentence structures); and germane load, referring to the reader's active effort in integrating the information (e.g., mentally summarizing content or making inferences).

Due to the internal nature of cognitive processes, directly measuring cognitive load is challenging. Various studies have attempted to infer it using physiological and behavioral signals collected during reading. Eye-tracking metrics (e.g., fixation duration, regressions, pupil dilation) [35, 87] and EEG signals (e.g., increased theta and reduced alpha activity) [39, 41] have been shown to correlate strongly with mental effort. Additional indicators such as heart rate variability [75] and electrodermal activity [6] also reflect sustained cognitive strain. These signals provide real-time, user-specific feedback and have been used in multimodal datasets like ZuCo [30] and CLARE [5] to train models for cognitive load prediction.

While these methods offer accurate measurement of cognitive load, they require specialized hardware and calibration, which limits their practicality in the context of human-LLM interaction. Therefore, we focus on metrics that can be directly calculated from the text content itself to infer cognitive load.

Text-Based Cognitive Load/Readability Estimation. Readability refers to the ease with which readers can comprehend written text and has been shown to correlate strongly with the cognitive load experienced by humans when processing textual content [10]. Traditionally, readability has been approximated using surface-level linguistic features such as sentence length, word length, and vocabulary frequency [12]. Classic text difficulty formulas, including Flesch-Kincaid [38], Dale-Chall [11], and Gunning Fog [28], provide fast estimates of readability but often fail to capture deeper semantic, syntactic, or discourse-related complexity [9]. To address these limitations, researchers have proposed syntactic complexity metrics [3] (e.g., clause density, parse tree depth) and cohesion indices (e.g., referential overlap, connectives) to more accurately reflect the processing effort required for comprehension. Tools such as Coh-Metrix [25] compute hundreds of such features to model cognitive load more comprehensively. While these methods are interpretable and efficient, they are limited by their reliance on static textual features and often fail to account for longer-range contextual factors.

Machine learning approaches have advanced the estimation of cognitive load during reading by learning directly from annotated datasets rather than relying solely on predefined rules. Early models combined linguistic features with classifiers or regressors to predict reading level or cognitive demand [22, 63]. More recent work leverages pre-trained transformer-based language models such as BERT [15] and GPT [65], fine-tuned on datasets including WeeBit [81], OneStopEnglish [80], and Newsela [85]. These models outperform traditional approaches by capturing rich syntactic and semantic patterns, although they often lack interpretability and require substantial labeled data. Despite these trade-offs, ML-based approaches offer scalable and accurate cognitive load estimation, making them well-suited for real-time applications. Recent studies further show that LLMs can provide general-purpose assessments of cognitive demands in longer and more complex paragraphs [51, 62]. By leveraging world knowledge and simulating human-like comprehension, LLMs offer a more integrated and context-aware framework for estimating cognitive load from text.

2.2 Resource Optimization for Streaming Output

Efficient LLM Serving. As cloud-based LLMs like ChatGPT scale to serve over a billion user interactions daily, the efficient delivery of AI-generated content has become a critical concern. Recent work has proposed system-level optimizations that treat streaming timing as a tunable parameter to improve both latency and resource utilization. For instance, Andes [52] introduces a Quality-of-Experience-aware scheduling system that reallocates GPU time by detecting when users accumulate unread tokens, temporarily pausing streams to better balance serving capacity across users. Similarly, AdaServe [49] leverages fine-grained speculative decoding to dynamically adjust token generation speed to improve serving efficiency. TimelyLLM [50] focuses on robotic applications of LLMs, efficiently optimizing the LLM serving schedule by leveraging the time redundancy between robot plan generation and execution. On the transmission layer, Eloquent [48] improves token delivery over the network by merging newly generated tokens and currently unacknowledged tokens in the next outgoing packet [83].

However, these methods focus primarily on system-level metrics and overlook the variability in LLM-generated content, which often differs in readability and affects how quickly users can consume the output.

Adaptive Text Delivery. Beyond LLM use cases, researchers have explored adaptive text delivery in other reading interfaces and intelligent tutoring systems. Early work like Time Aura [54] introduced ambient visual pacing interfaces that helped users regulate their pace during cognitively demanding tasks such as test-taking or presentations. Lander et al. [46] developed gaze-adaptive scrolling that uses eye-tracking to adjust the pace of text scrolling for public display. In immersive environments, Grootjen et al. [26] presented a VR system that adapts word speed based on pupil dilation, a proxy for cognitive load, to slow down when users are overwhelmed and speeding up when they are not. Intelligent Tutoring Systems like AutoTutor [24] and Gaze Tutor [16] similarly adjust the pacing of explanations or interventions based on learner engagement, gaze, or confusion.

Another extensively explored technique for adaptive text delivery is Rapid Serial Visual Presentation (RSVP), where individual words are displayed sequentially at a fixed screen location. Early pioneering work by Öquist et al.[59, 60] introduced adaptive RSVP algorithms that modulate presentation timing based on textual features, achieving reduced task load compared to static RSVP. More recently, Kosch et al.[40] incorporated EEG-based cognitive load monitoring and found significant correlations between physiological signals and presentation speed, proposing dynamically adjusted RSVP rates based on EEG signals.

Other Scenarios. Adaptive streaming techniques have also been applied in multimodal interfaces other than text. For example, in multimedia classroom, segmentation of instructional videos and adaptive pausing based on gaze or confusion signals has been shown to improve comprehension and retention [32, 33, 45]. In smartphone use, attention-aware systems like Attelia [57] detect cognitive breakpoints in smartphone usage and delay notifications until moments of low cognitive load, reducing disruption and information overload. In narrative visualizations, gaze-driven systems dynamically highlight relevant content as users read, enhancing the coupling between text and data [2, 44]. In VR, human perception of color has been utilized to adjust the streaming display quality to reduce the power consumption [18].

These examples show how user attention and perceptual constraints can guide adaptive pacing. The principles behind these systems—detecting cognitive state and dynamically adjusting delivery pace—can inform the design of text streaming systems in LLMs.

3 What is the Benefit of Adaptive Streaming?

We begin by defining the scope of our study. First, we focus on scenarios involving human–LLM interaction, where users actively consume the generated content as it is streamed. Other applications of LLM, such as code generation or agent–agent communication, fall outside the scope of this work.

Second, we consider a cloud-based LLM serving environment, where computational resources are shared among multiple users simultaneously. We do not aim to optimize the experience of single-user scenarios, such as locally deployed LLMs.

Last, we adopt a statistical perspective to model and analyze the problem. Our objective is to optimize the aggregated user experience under a given computational budget, rather than tailoring the streaming speed to specific users.

With this scope in mind, the central research question we aim to answer is: **How much computing resource can be saved by adaptively modulating text streaming speed?** Namely, under what conditions can adaptive streaming improve computational efficiency, and to what extent would it affect overall user experience? Answering these questions helps justify the need for our system. In this section, we develop a statistical framework to explore and quantify these aspects.

3.1 Quantifying Computing Resource Savings

Intuitively, content with low information density, simple sentence structures, and familiar vocabulary is easier to read and thus requires a faster streaming speed—streaming too slowly in these disrupts the user's reading flow and degrades their experience. More complex and information-dense content naturally slows down users' reading pace, allowing for slower streaming without negative impact. Moreover, streaming content faster than a user's reading speed does not enhance the experience either [1], since users will naturally read at their own pace regardless of how quickly the text becomes available. This means any excess delivery speed is effectively unused and instead leads to inefficient use of system resources. This asymmetry motivates an adaptive streaming approach that aligns output speed with the user's expected reading pace.

Modeling Reading Speed. Without loss of generality, we consider a setting in which two groups of users interact with a cloud-based LLM service, where each group consumes generated content with distinct cognitive demands—arising from differences in topic, writing style, or text complexity (e.g., Group A reads narrative stories, while Group B reads technical papers).

Although reading speed varies across individuals due to factors such as personal habits, familiarity with the content, cognitive processing ability, and language proficiency, prior research suggests that users' dwell time on online articles follows a log-normal distribution due to its non-negative and right-skewed nature [86]. Therefore, for each group, we model the user's natural reading speed as a log-normal distribution,

$$r_A \sim \text{Lognormal}(\mu_A, \sigma_A^2),$$
$$r_B \sim \text{Lognormal}(\mu_B, \sigma_B^2),$$

where μ_A and μ_B are the means of the underlying normal distributions (i.e., $\log r$), and σ_A, σ_B are the corresponding standard deviations. Here, r represents the natural reading speed when the entire passage is fully presented (i.e., not streamed).

Our core assumption is that the streaming speed should be at least equal to a user's natural reading speed to prevent noticeable delays in content delivery. That is, for a given group, if we stream at speed s such that

$$\Pr(r \le s) \ge \alpha,$$

then a fraction α of users will receive content at a pace that matches or exceeds their reading speed, while the rest may notice latency due to slower-than-expected delivery. This defines the *Streaming-Reading Alignment Rate* (SRAR), denoted as α, which represents the fraction of users whose natural reading speed is less than or equal to the chosen streaming speed s.

Our goal here is to save computational resources while maintaining a high target SRAR across all users.

Savings When Resources Are Sufficient. Given the log-normal distribution of reading speeds, the α-quantile (i.e., the minimum speed that covers α percent of users) can be computed using the inverse Cumulative Distribution Function of the log-normal distribution:

$$s_A^{(\alpha)} = \exp\left(\mu_A + z_\alpha \cdot \sigma_A\right),$$
$$s_B^{(\alpha)} = \exp\left(\mu_B + z_\alpha \cdot \sigma_B\right),$$

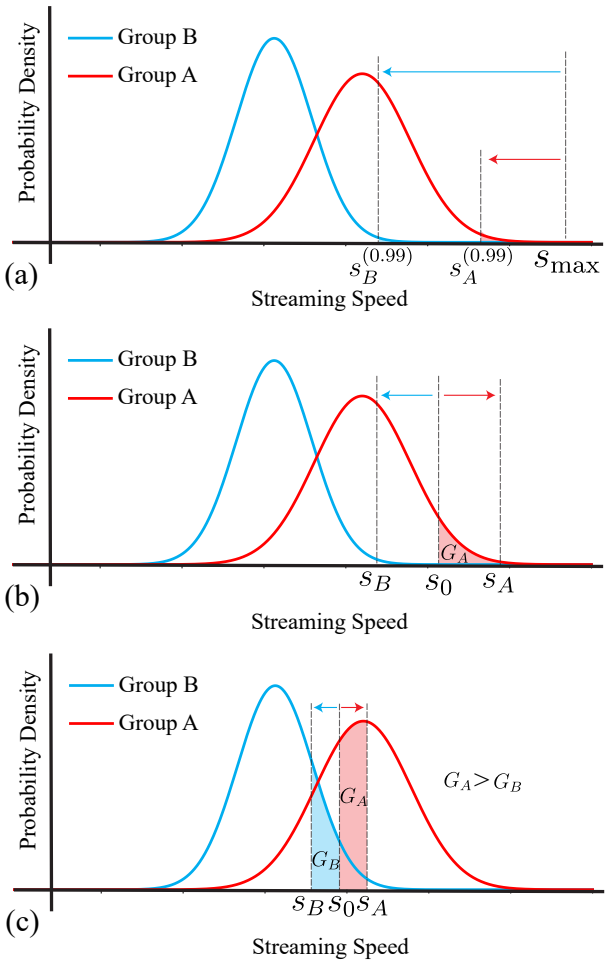

Figure 3: Potential computing resource savings of adaptive streaming under different scenarios. Arrows indicate streaming speed adjustments. (a) When the maximum LLM streaming speed $S_{\max}$ is very high, it can be reduced to $s_A^{(0.99)}$ and $s_B^{(0.99)}$ for Groups A (easier content) and B (harder content), respectively. This maintains a comfortable reading speed for 99% of users in each group while significantly reducing resource usage. (b) When the available resource can only support both Groups A and B at a streaming speed of s_0, which is lower than the thresholds in (a), it is still possible to reduce Group B's speed to s_B and reallocate the saved resource to increase Group A's speed. This enhances Streaming-Reading Alignment Rate (SRAR) for Group A (area G_A) with minimal impact on Group B's experience. (c) If the available computing resources is even more constrained, a trade-off can still possibly improve overall SRAR. By reducing Group B's speed and increasing Group A's speed, overall SRAR improves as long as $G_A > G_B$, resulting in a net gain of $G_A - G_B$.

where z_α is the z-score of a normal distribution corresponding to percentile α (e.g., $z_{0.99} \approx 2.33$, $z_{0.95} \approx 1.64$, and $z_{0.90} \approx 1.28$).

Let $s_{\max}$ denote the maximum streaming speed supported by the system (i.e., streaming at the full generation rate without throttling). In traditional non-adaptive streaming, when resources are sufficient, text is streamed at $s_{\max}$ to all users, regardless of content.

In contrast, our adaptive streaming allows the rate to be reduced to $s_A^{(\alpha)}$ and $s_B^{(\alpha)}$ for each group, while still ensuring a desired quality threshold α. Let p_A and p_B be the proportions of users in Groups A and B, respectively, such that $p_A + p_B = 1$. Then the expected computing resource saving under adaptive streaming is

$$\text{Saving}(\alpha) = 1 - \frac{p_A s_A^{(\alpha)} + p_B s_B^{(\alpha)}}{s_{\max}}. \tag{1}$$

This provides a tunable control knob of a trade-off: higher values of α ensure a better user experience but yield less savings, while lower α values improve efficiency at the cost of allowing more users to experience slower streaming. Figure 3(a) illustrates the potential resource savings under this scheme. Suppose we aim to stream at a rate that exceeds the reading speed of 99% of users in each group. In that case, we can set the streaming speeds to $s_A^{(0.99)}$ and $s_B^{(0.99)}$ for Groups A and B, respectively. The corresponding resource savings can then be computed using Equation 1.

Savings When Resources Are Limited. We have characterized the potential computing resource savings when the system's maximum streaming speed is sufficiently high. However, in practice, the LLM server may not have enough capacity to stream at those speeds. Suppose the available resources only permit an averaged streaming rate of s_0 for both groups, and assume equal group sizes, i.e., $p_A = p_B = 0.5$.

In this constrained scenario, shown in Figure 3(b), a non-adaptive strategy would be to stream at s_0 uniformly, which will result in a certain number of user-observed delays. Instead, we can redistribute resource by slightly decreasing Group B's speed to s_B, which still maintains a high SRAR in that group, and reallocate the saved resource to increase Group A's speed to s_A. This adjustment enables a new subset of Group A users—denoted as area G_A—to receive content at or above their natural reading speed, improving their experience without largely degrading the experience for Group B.

In an even more resource-limited case, illustrated in Figure 3(c), the shared rate s_0 is further reduced. Here, lowering Group B's speed to s_B will result in more users in Group B falling below their comfortable reading threshold (represented as area G_B), but the reallocated resource allows a larger number of Group A users (area G_A) to be covered. If the goal is to maximize overall user SRAR when average speed is limited at s_0, this redistribution is beneficial when $G_A > G_B$. If $G_A < G_B$, such move would reduce the overall SRAR and should be avoided.

Overall, when resources are limited, adaptive reallocation of streaming speed can still improve user experience, but must be guided by the marginal gains and losses across groups.

3.2 A Real Example from Crowdsourcing User Study

The previous section introduced a theoretical framework for analyzing potential computing resource savings and balancing the trade-off between efficiency and user experience. Here, we present

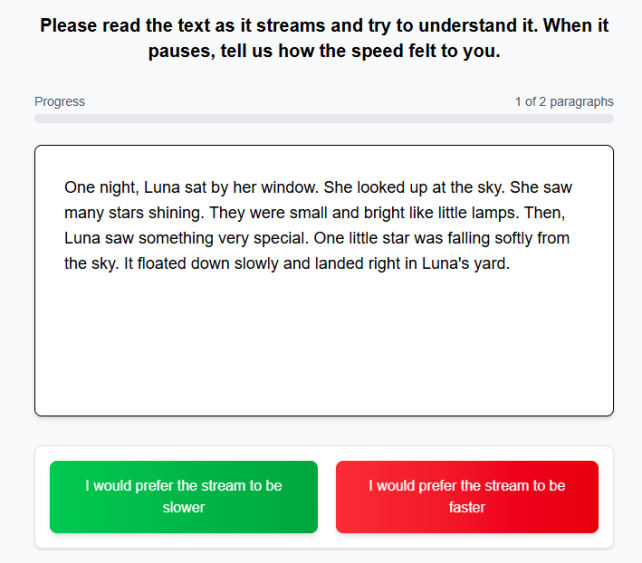

Figure 4: A screenshot of our crowdsourcing web interface.

a real-world text streaming scenario through a crowdsourced user study, offering concrete empirical results. This not only helps validate key theoretical assumptions but also provides practical examples for those looking to apply a similar adaptive streaming approach in their own LLM systems.

Study Protocol. In this test, we conducted a user study based on two distinct content scenarios: (1) a bedtime story suitable for a six-year-old child, and (2) an explanatory passage on an economic concept. These texts were generated offline using GPT-4o to simulate content streamed by LLM service. We used the Flesch–Kincaid readability tests [38] to verify that the readability level of the bedtime story is below 5th grade, while the economic passage is at the college level. We also explicitly prompt GPT-4o to generate content that maintains sentence-level consistency within each passage.

We recruited 100 participants through the Prolific platform, as prior research suggests that Prolific provides higher-quality data than other crowdsourcing platforms, and sometimes even better than in-person studies [17]. Participants ranged in age from 21 to 68 years (avg. 34.7), with 57 male and 43 female. Additionally, 58% of participants held a bachelor's degree or higher. All participants were required to be native English speakers to eliminate the influence of language proficiency. Each participant was asked to read two passages—one from each content type—using a simulated streaming interface (see Figure 4). Participants were compensated $3 USD for completing the 5-minute study. The study protocol was reviewed and approved by our internal legal and ethics committee.

The interface implements a Parameter Estimation by Sequential Testing (PEST) [78] procedure to estimate each participant's natural reading speed. Participants were instructed to "Read the text as it streams and try to understand it. When it pauses, tell us how the speed felt to you." At each pause, participants could indicate whether they preferred the speed to be faster or slower. Based on their response, the streaming continued, and the speed was adjusted by increasing (if chose "prefer faster") or reducing (if chose "prefer slower") Δv. The value of Δv was also updated after each selection according to the standard PEST update rule.

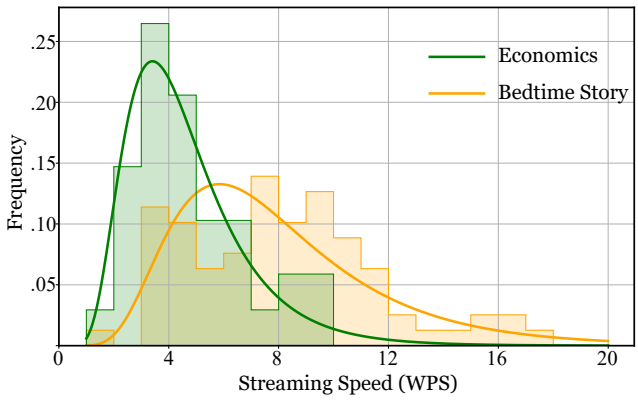

Figure 5: Histogram of preferred streaming speeds (in words per second) from the crowdsourced user study, overlaid with fitted log-normal distributions. The green curve represents the Economics content, while the orange curve corresponds to the Bedtime Story content.

$$\Delta v = \begin{cases} \max\left(\Delta v_{\text{last}}/2,\ 0.2\right), & \text{if a choice reversal occurs} \\ \Delta v_{\text{last}}, & \text{if the choice is consistent} \end{cases}$$

The initial streaming speed was set randomly from 3 to 8 words per second (WPS), with an initial Δv of 2 WPS.

After seven speed adjustments, participants were presented with an additional option: "This is the same as my reading speed." They could either accept this speed or continue adjusting. The final selected speed was recorded as the participant's comfortable reading speed for the corresponding passage.

In psycholinguistic research [55], similar self-paced reading methods are widely used to assess reading speed. In these tasks, participants read text on a computer screen and press a key to reveal each new segment, with the reading rate derived from their response times. However, because our goal is to more closely replicate the continuous streaming scenario typical of LLM interactions, we designed our system to stream a paragraph at a fixed speed and used the PEST procedure to determine each participant's comfortable reading speed instead.

To ensure data quality, we employed control checks to identify inattentive or inconsistent behavior. Participants were flagged for exclusion if they (a) consistently selected "prefer faster" regardless of the content, or (b) reported a faster preferred speed for the more complex (economics) passage than for the simpler (bedtime story) passage. The rationale for these criteria is twofold: (a) after seven consecutive "faster" selections, the speed would increase to approximately 20 WPS (1,200 WPM), which is well beyond the human reading limit even for the fastest readers; and (b) it is rare for someone to read a college-level text faster than a fifth-grade passage. In total, 21 out of 100 participants failed at least one of these control checks and were excluded from the analysis, which aligns with the 22.3% failure rate reported in a previous study on crowd workers [69].

Results. We treat the final selected speed as the comfortable reading speed for each participant on each passage, and fit a log-normal distribution to the collection of these speeds. Figure 5 shows the histogram of the comfortable streaming speeds alongside the fitted log-normal curves. The green bars and curve correspond to the more complex economics passage, while the orange bars and curve represent the simpler bedtime story. The peaks of both the histograms and the fitted distributions indicate that the comfortable reading speed for the complex passage is generally lower than that for the simpler one.

We further validated this difference using a paired t-test, which revealed a statistically significant result: $t(78) = 11.81$, $p < 0.001$. (78 is degrees of freedom correspond to 79 effective paired samples.) To evaluate the quality of the log-normal fit, we conducted a Kolmogorov-Smirnov (K-S) test [4]. For both passages, the K-S statistic was 0.10, indicating a good alignment with the log-normal model. The corresponding p-values (0.35 and 0.39, respectively) suggest that the observed distributions do not significantly deviate from the log-normal assumption.

Using the log-normal distributions of reading speed inferred from the crowd-sourced data, we can compute the potential resource savings achieved through adaptive streaming for the two passages. First, we estimate the savings in a scenario where computing resources are sufficient. We assume $s_{\max}$ to be the average streaming speed of the GPT-4o API, which is approximately 45 WPS second [29]. The 99th percentile of the log-normal distribution is 21.20 WPS for the simpler passages and 11.97 WPS for the more complex one. By streaming at these speeds and using Equation 1 for computation, we can achieve a **63.14%** reduction in computing resource usage while maintaining a 99% SRAR.

Even with limited computing resources, similar to the scenario in Figure 3(c), adaptive streaming can improve overall SRAR. The intersection of the two log-normal curves occurs at 5.53 WPS, suggesting that as long as the average computing budget is at least 5.53 WPS, adaptive streaming can lead to a net gain in user experience.

4 System Implementation

Section 3 presented a statistical framework and experimental procedure that help understand the potential benefits of adaptive streaming. In that analysis, the distribution of the reading speed was obtained through an offline user study. But in a real-time LLM system, the content to be streamed is unknown until a user request is received. As a result, the corresponding reading speed distribution also cannot be determined in advance.

Prior studies have shown that human reading speed is largely influenced by the cognitive load imposed by the content [14, 21]. Building on this insight, a practical adaptive streaming system can use estimated cognitive load as a primary signal for determining appropriate streaming speed. To enable real-time adjustments, the system must estimate the cognitive load of each segment as it is being streamed and ensure that the streaming speed is allocated accordingly.

This section describes our system prototype, which includes a cognitive load estimation method and a resource allocation algorithm for adjusting streaming speed under budget constraints.

4.1 Cognitive Load Estimation

Section 2.1 discussed several cognitive load estimation methods for textual content, which span a wide range, from simple linguistic features that can be computed almost instantaneously, to more advanced machine learning models, or even hardware-based approaches. While hardware-based methods provide the most accurate measurement of cognitive load, they are not feasible for use in the client end of an cloud-based LLM service.

In our prototype system, we explore two different cognitive load estimators. First, we use the Gunning-Fog Index [28], a widely used readability metric that approximates the cognitive load of a text paragraph. Due to its simplicity, it can be computed with negligible overhead, making it suitable as a lightweight plugin for integration into LLM-based systems.

The second estimator leverages the LLM itself to assess cognitive load, inspired by the LLM's strong performance on subjective, human-like evaluation tasks [27]. Specifically, we provide the LLM with a definition of cognitive load based on Cognitive Load Theory and prompt it to append a cognitive load score using a special tag symbol, <X>, after generating a text segment. The number inside the brackets (e.g., <3>) represents the estimated cognitive load on a fixed scale, where lower values correspond to lower expected reading speeds, indicating more complex or cognitively demanding content. We constrain the score to a range from 1 to 10, where 1 denotes the highest cognitive load.

Example of Generated Text:

> By introducing the concept of spacetime, the theory of relativity fundamentally transformed the landscape of modern physics. <3> It unified space and time...

During streaming, the system detects the tag symbol and extracts the inferred cognitive load score without displaying it to the user. This method is particularly useful because it requires no additional external models and only adds the computation of a few tokens per segment. It can be seamlessly integrated into any existing LLM system by simply modifying the prompt instructions, making it a practical and scalable solution for real-time cognitive load estimation.

4.2 Resource Allocation and Speed Modulation

We formulate our resource allocation problem as follows. Suppose at a given time, the server is processing n concurrent user requests, each corresponding to a distinct text segment to be streamed. Let the total available computational capacity at this moment be sufficient to stream k words within a unit time interval. Without any speed modulation, each of the n segments would be streamed uniformly at a speed of k/n words per unit time.

To better align streaming speeds with estimated user cognitive load, we introduce adaptive speed modulation method. Specifically, let s_i denote the cognitive load score for the i-th segment among all n concurrent request. Our goal is to stream more cognitively demanding (lower-scoring) segments at slower speeds, providing users additional processing time, while allocating faster streaming speeds to segments with lower cognitive complexity (higher scores).

To achieve this, we first normalize the cognitive load scores into allocation weights (s_i) that sum to 1. We then apply a smoothed interpolation between a linear distribution of weights and a uniform distribution:

$$w_i = \alpha \cdot \frac{s_i}{\sum_{j=1}^{n} s_j} + (1-\alpha) \cdot \frac{1}{n}, \quad i = 1, 2, \ldots, n \tag{2}$$

Here, $\alpha \in [0, 1]$ is a hyperparameter that controls the interpolation between a purely linear weighting (based on normalized cognitive load scores) and a uniform weighting (equal speed for all segments). When $\alpha = 1$, the weights are fully proportional to the cognitive load scores; when $\alpha = 0$, all segments receive equal weights regardless of content complexity. The choice of α depends on the desired sensitivity of streaming speed to content complexity and can be determined empirically through a small-scale user study or by analyzing prior usage data. In our implementation, we set $\alpha = 0.5$ for both the Gunning-Fog estimator and the LLM-based estimator, representing a balanced interpolation.

With these weights, the streaming speed v_i (in words per unit time) allocated to each segment is computed as:

$$v_i = w_i \cdot k, \quad i = 1, 2, \ldots, n$$

This adaptive speed modulation algorithm ensures that computational resources are allocated in proportion to the cognitive complexity of each segment. The tunable parameter α provides both flexibility and interpretability to the resource allocation mechanism.

4.3 Other Practical Considerations

Backtracking Speed Control Token. It is important to note that our current approach calculates the streaming speed allocation only after the server has already generated a given text segment. As a result, we cannot retroactively slow down the generation of that segment on the server side. A practical implementation of streaming speed control, therefore, is to let the client side stream the newly generated text at the determined speed, while the server pauses or adjusts the generation of subsequent sentences to align with the updated allocation. This ensures that users experience streaming at the calculated speeds in real time. Implicitly, this approach assumes that adjacent sentences tend to have similar cognitive load. This idea connects to the notion of local coherence, a well-established concept in natural language processing that suggests neighboring sentences are systematically related in aspects such as sentence length, discourse structure, and lexical or topical continuity [36, 70], which are factors that associated with the complexity of the textual content [13, 71].

Modeling Computational Resource. In our theoretical framework above, for simplicity, we treated generation speed as the primary computational resource to manage and allocate. However, in practical deployment, there are multiple ways to define and measure computational resources, including total cost budgets, available FLOPS [7], the number of GPUs, or total power consumption. It is important to note that, depending on which resource model is used, the mechanism of speed modulation, and the relationship between computational efficiency and user experience can vary.

To best align our analytical model with practical implementation, one practical perspective is to treat power consumption as the resource budget being managed, with GPU clock frequency serving as the tunable parameter. In this approach, increasing the clock frequency results in faster model inference and higher power usage, while reducing the frequency slows down inference and conserves power. This technique, known as Dynamic Voltage and Frequency Scaling (DVFS) [47, 56, 77], is widely used in resource management for both CPUs and GPUs. On NVIDIA GPUs, power management can be achieved through the `nvidia-smi` API. Recent research also demonstrates a concrete example of dynamically tuning GPU frequency for efficient LLM serving [37]. By tuning the clock frequency, a service provider can adjust generation speed with a single control parameter, without requiring additional operations such as memory migration or task scheduling, which can incur extra overhead.

Speed Allocation Assumption. Our speed allocation scheme (Equation 2) relies on one key underlying assumption: that under a fixed computational budget, streaming speed can be allocated linearly across different user requests. That is, decreasing streaming speed by Δs for one request allows the provider to increase streaming speed by Δs for another request. However, in real-world industry practice, the relationship between LLM inference speed and power consumption is complex and evolving, involving factors such as GPU types, LLM model architecture, and batch size. Given this complexity, modeling this phenomenon precisely is challenging without using simplifying assumptions.

Prior research suggests that GPU clock frequency is monotonically and positively correlated with LLM inference speed and power consumption [37, 53]. This positive correlation still supports the validity of our method: reducing the speed for more complex text can free up resources to increase speed for simpler text, thereby improving SRAR, even if the trade-off may not be exactly one-to-one. Additionally, one recent investigation [53] and an industry report from the NVIDIA GTC 2025 Keynote [84] found that within certain ranges, power consumption may scale linearly with LLM inference speed. We emphasize that detailed modeling of these effects is complex and beyond the scope of this study. For tractability, we adopt the simplified linear allocation assumption throughout all our analyses and simulations, with the goal of providing theoretical insights that can inform the practical design of our adaptive streaming method.

5 Crowdsourcing Experiment

In this study, we aim to address two central research questions regarding our proposed adaptive streaming system: First, how effectively does our cognitive load estimation method perform under realistic usage scenarios? Second, how does our adaptive streaming approach compare to the baseline method of streaming content at a uniform speed, in terms of computational resource efficiency and SRAR?

To investigate these questions, our system combines cognitive load estimation with the adaptive speed modulation algorithm described in Section 4.2 to determine appropriate streaming speeds for different text passages under varying computational resource constraints. To provide empirical grounding, we conducted a crowdsourcing study to obtain distributions of users' comfortable reading speeds of the passages. Once these distributions are collected and

streaming speeds assigned, we compute the overall SRAR—defined as the proportion of users who receive streaming speeds equal to or greater than their natural reading speeds (recall the definition at Section 3)—at each level of computational budget, for all speed modulation methods. Finally, we simulate and compare the computational resources required by each method to achieve equivalent SRAR.

5.1 Data Collection

The crowdsource user study conducted here follows a procedure similar to the one described previously in Section 3.2. To construct a diverse dataset, we first prompted GPT-4o to generate 10 English text passages spanning various randomly selected topics. We also explicitly instructed the LLM to produce content of varying cognitive loads across passages. For calibration, the simplest content is set as a bedtime story suitable for a six-year-old child, while the most complex content is an explanatory passage about a philosophical concept. All other topics and corresponding passages were generated autonomously by the LLM within this defined complexity range. To ensure quality, we verified that each passage could be understood without requiring specific technical knowledge, with lengths ranging consistently between 150 and 200 words.

We recruited 200 native English speakers through the Prolific platform. Participants consisted of 44% male and 56% female, with an average age of 38.6 years (ranging from 24 to 74). Among participants, 57% reported holding a bachelor's degree or higher. During the study, each participant was presented with 6 randomly selected passages using the same simulated streaming interface and PEST-based speed adjustment procedure described in Section 3.2. The passage order was fully randomized across participants to avoid order effects. Each session lasted about 15 minutes to reduce fatigue and minimize fluctuations in reading speed, and each participant was paid $7 USD for the study.

We applied similar control checks as described in Section 3.2 to ensure data quality. Specifically, participants were flagged if they selected "faster" for every adjustment within a passage or if their preferred reading speed for the most complex passage was faster than for the easiest passage. In addition to these checks, after finishing each passage, the passage would disappear, and we then asked a two-choice comprehension question, generated by the LLM, to verify whether participants understood the content. For example, one passage tells a short story about a child baking cookies with their grandmother, and the corresponding question is: "What food is the child trying to make?" All control checks needed to be satisfied for a data point to be considered valid. After applying these quality checks, we ensured that each passage received between 80 and 100 valid data points representing individual users' comfortable streaming speeds.

5.2 Results and Analysis

Validation of Cognitive Load Estimation Methods. To first evaluate the quality of our cognitive load estimation approaches, we examined their correlation with empirically measured comfortable reading speeds. Specifically, we calculated Pearson's correlation coefficients between each method's cognitive load scores and the median comfortable reading speeds obtained from our crowdsourcing experiment for each passage. The results demonstrated strong and statistically significant correlations for both methods: the LLM-based cognitive load estimation achieved a correlation of ($r = 0.955$, $p < 0.001$), whereas the Gunning-Fog Index yielded a correlation of ($r = 0.828$, $p = 0.003$). These results confirm that both cognitive load estimation methods effectively approximate users' comfortable reading speeds, with the LLM-based approach exhibiting notably higher predictive accuracy.

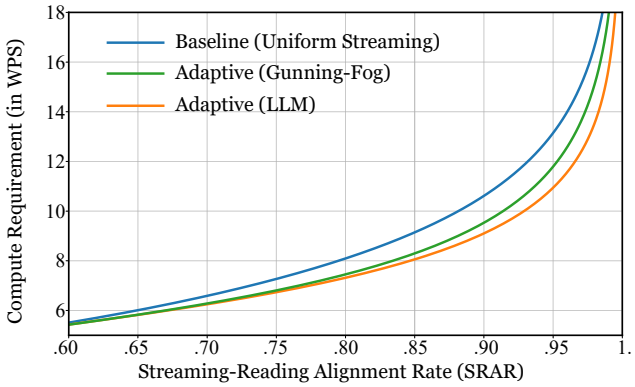

Figure 6: Compute requirements of different methods at varying SRAR. The compute requirements are measured by average streaming speed in words per second (WPS), where lower is better. At the same overall SRAR, our proposed methods (Gunning-Fog and LLM) consistently show lower compute requirements compared to the baseline.

Comparison of Resource Efficiency and SRAR. To better illustrate the overall effectiveness of cognitive-aware resource allocation, we plotted the resource utilization of different methods across varying SRAR levels in Figure 6. The x-axis represents the SRAR, while the y-axis indicates computational resource usage, defined as the average streaming speed (in WPS) across all concurrently streamed passages required to achieve each SRAR level. Lower values on the y-axis indicate greater resource efficiency.

As shown in Figure 6, at equivalent SRAR, the Gunning-Fog-based allocation method (green line) consistently requires fewer computational resources compared to the uniform-speed baseline (blue line). Moreover, the LLM-based allocation method (orange line) yields even greater efficiency gains. Notably, at a high SRAR target of 95%, the Gunning-Fog-based method achieves a 10.33% reduction in computing resources relative to the baseline, while the LLM-based adaptive modulation attains an even more substantial reduction of 16.79%. Table 1 presents the exact average streaming speed required by each method to achieve the target SRAR, as well as the compute savings ratio of our proposed methods compared to uniform streaming.

These findings highlight clear efficiency improvements achieved through adaptive streaming. The Gunning-Fog Index, as a straightforward off-the-shelf metric, incurs negligible computational cost while still providing a meaningful increase in streaming efficiency. On the other hand, the LLM-based estimation method, which produces cognitive load scores more closely aligned with actual human

SRAR	Baseline	Gunning-Fog		LLM	
	Compute	Compute	Save %	Compute	Save %
0.65	6.01	5.83	3.00%	5.82	3.16%
0.70	6.59	6.28	4.70%	6.25	5.16%
0.75	7.27	6.81	6.33%	6.74	7.29%
0.80	8.10	7.46	7.90%	7.32	9.63%
0.85	9.15	8.30	9.29%	8.06	11.91%
0.90	10.62	9.54	10.17%	9.11	14.22%
0.95	13.16	11.80	10.33%	10.95	16.79%

Table 1: Reduction in computational resource usage (%) achieved by cognitive load estimators compared to the uniform streaming baseline at varying SRAR. "Compute" is measured as average streaming speed in WPS, with lower values indicating lower resource usage.

reading speeds, achieves even greater overall efficiency improvements.

We also observe that when the streaming speed is lower than 6 WPS, there is little difference between uniform streaming and adaptive streaming. This corresponds to the scenario illustrated in Figure 3(c), where adaptive streaming may result in more users experiencing delays. The value of 6 WPS provides an empirical insight from real data into the threshold at which the benefits of adaptive streaming begin to diminish.

6 Applications, Limitations, and Future Directions

While our framework demonstrates promising results, there remain several limitations and opportunities for further development across application scenarios, model design, and system-level optimization.

Application Scenario. In principle, our adaptive streaming framework requires providers to deploy LLMs on their own GPU infrastructure to control generation speed. Such providers includes major services such as ChatGPT, Google Gemini, Claude, and DeepSeek, as well as any organization deploying open-source LLMs on their own infrastructure. Nowadays, many services developed by startups or individual developers are powered by open-source LLMs that are fine-tuned for specific conversational AI scenarios [82], such as customer support, educational chatbots, or personal assistants. Unlike large companies, these providers often have more limited GPU resources. Employing our method could help them make better use of their resources and enable higher service bandwidth.

We also acknowledge that certain LLM use cases may not benefit directly from adaptive streaming based on cognitive load. Specifically, tasks that utilize LLM outputs primarily as intermediate computational steps, such as code generation, data processing pipelines, or internal reasoning procedures, typically prioritize rapid delivery with users focusing primarily on the final results rather than incremental outputs. In these contexts, users prefer the model to stream tokens at the maximum available speed, making cognitive-aware streaming less relevant.

Nevertheless, our adaptive streaming mechanism remains highly applicable to general-purpose conversational AI through the integration of an intention detection module following the user's prompt. For instance, when users request explanatory content ("Explain the theory of relativity"), instructional guidance ("How to make pasta dough"), or in-depth informational answers ("What are the symptoms and treatments for diabetes?"), the intention detector identifies these interactions as reading-intensive tasks suitable for adaptive streaming. Conversely, task-oriented interactions (e.g., "Generate Python code for sorting algorithms") bypass adaptive streaming, allowing for immediate, maximum-speed token delivery and efficient resource allocation.

Moreover, there has been extensive recent research and development of LLM-enabled interactive conversational systems designed to assist users through detailed textual interactions. Representative applications include AI-powered educational tutors delivering personalized learning content [89], virtual healthcare assistants providing medical information and instructions [66], emotional support chatbots [88], and physical activity coaching systems [34]. In these scenarios, users typically engage deeply with the content and often ask follow-up questions. Deploying our cognitive-aware adaptive streaming framework in such applications could help these systems serve more users simultaneously while maintaining a high-quality user experience. By aligning streaming rates with users' processing capabilities, our approach directly contributes to ongoing efforts in HCI to design more responsive, scalable, and resource-efficient conversational AI services.

Cognitive Load/Reading Speed Prediction. As a prototype implementation, we currently use linguistic features and the LLM itself to predict cognitive load. While this approach demonstrates some effectiveness, it may not precisely model the user's cognitive load on every aspect. One direction for future work is to develop a model that can make fast predictions while also estimating the distribution of reading speeds more accurately, thereby supporting more efficient resource allocation.

Another promising avenue is adapting the system to specific application scenarios. For instance, if the LLM is deployed to provide information about a particular product to customers, the service could analyze users' historical requests alongside the corresponding LLM-generated responses. This data could then be leveraged to model the distribution of users' reading speeds. Given the specific focus of such applications, the estimation could be accurately performed with a small-scale user study. The resulting distribution would be useful in optimizing resource allocation for that application in the future.

Additionally, emerging AR/VR headsets typically include gaze-tracking capabilities, which could be used to directly measure users' reading speeds. For AI assistant applications deployed on AR/VR headsets, our adaptive streaming technique could leverage gaze tracking data to dynamically adjust the streaming rate, ensuring that the content delivery closely matches the user's consumption speed.

Resource Allocation with Low-level Mechanism. In our current study, we treat computational resource allocation as a speed bandwidth optimization problem. This can be practically implemented using DVFS, which enables us to evaluate our adaptive streaming method in a controlled environment and analyze its performance. In real-world deployments, resource optimization may also involve additional factors related to low-level system mechanisms such as

data transmission, hardware constraints, and cache management. In future work, it would be valuable to incorporate these factors and explore joint optimization strategies and integrate our method with existing serving libraries such as vLLM [43], to further enhance streaming efficiency.

Exploring Different Speed Adjusting Mechanism. Currently, our method uses speed modulation to align with users' reading speeds. However, users may engage with content in different ways. For example, after reading logically complex content such as mathematical formulas, users may pause and reflect before continuing. Inspired by this observation, it would be interesting to explore alternative streaming strategies, such as streaming at full speed followed by brief pauses, or combining variable speed modulation with pause mechanisms. Investigating how different streaming strategies affect user experience represents a compelling direction for future research.

Diversifying User Experience Metrics. Our current framework models the optimization problem with the goal of maintaining an overall SRAR, which is based primarily on statistical modeling and simulations. However, broader measures of user experience, such as perceived comfort, engagement, subjective workload ratings, and qualitative feedback, could offer richer insights. For example, our current approach does not explicitly factor in individual users' waiting times; it only considers whether the streaming speed falls below a user's comfortable threshold, without considering how far below that threshold the speed actually is. In the future, we could incorporate a weighted metric based on the magnitude of the difference between the actual streaming speed and the user's comfortable speed. Optimizing the system using such refined metrics could help achieve a more fine-grained balance between resource efficiency and user experience.

Distribution of Readers. Our current analytical framework assumes that reading speeds within a user group follow a log-normal distribution. This assumption has been empirically validated by our experiment using a K-S test, as described in Section 3.2. However, modeling human reading speed for any specific population remains a complex challenge, as it must account for diverse conditions such as dyslexia [72] or unusually fast readers [67], who may process information in fundamentally different ways. The characteristics of the user population can also vary significantly depending on the specific application scenario. In this paper, our goal is to provide a technical framework for implementing such a system, along with an example approach for evaluating its effectiveness, rather than offering a comprehensive study of human reading behavior. We hope this framework serves as a practical example that practitioners can adapt to analyze performance in their own settings.

7 Conclusion

To summarize, we propose the first method that leverages the cognitive load imposed on users to optimize the efficiency of LLM serving. While LLMs can stream tokens at high speeds, the output is not instantaneous—making the streaming rate a non-negligible factor in user interaction. Our adaptive streaming approach uses cognitive load signals to dynamically adjust token delivery speed, improving computational efficiency without largely impacting the user experience. Our crowdsourced user study demonstrates that this method increases the proportion of users whose streaming experience aligns with their content-dependent reading pace.

We hope this work encourages further exploration of user-centered and resource-aware LLM serving strategies, and inspires future research at the intersection of cognitive modeling, system optimization, and interactive AI design.

References

[1] Dina Acklin and Megan H Papesh. 2017. Modern speed-reading apps do not foster reading comprehension. *The American journal of psychology* 130, 2 (2017), 183–199.

[2] Oswald Barral, Sébastien Lallé, and Cristina Conati. 2020. Understanding the effectiveness of adaptive guidance for narrative visualization: a gaze-based analysis. In *Proceedings of the 25th international conference on intelligent user interfaces*. 1–9.

[3] Victor R. Basili and David H. Hutchens. 1983. An empirical study of a syntactic complexity family. *IEEE Transactions on Software Engineering* 6 (1983), 664–672.

[4] Vance W Berger and YanYan Zhou. 2014. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online* (2014).

[5] Anubhav Bhatti, Prithila Angkan, Behnam Behinaein, Zunayed Mahmud, Dirk Rodenburg, Heather Braund, P James Mclellan, Aaron Ruberto, Geoffery Harrison, Daryl Wilson, et al. 2024. CLARE: Cognitive Load Assessment in REaltime with Multimodal Data. *arXiv preprint arXiv:2404.17098* (2024).

[6] Mikołaj Buchwald, Szymo Kupiński, Adam Bykowski, Joanna Marcinkowska, Dawid Ratajczyk, and Marcin Jukiewicz. 2019. Electrodermal activity as a measure of cognitive load: A methodological approach. In *2019 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*. IEEE, 175–179.

[7] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. 2023. Run, don't walk: chasing higher FLOPS for faster neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12021–12031.

[8] Gautam Chutani. 2024. Streaming LLM Responses: Importance and Implementation. https://gautam75.medium.com/streaming-llm-responses-importance-and-implementation-911b135ef541 Accessed: 2025-03-30.

[9] Scott Crossley, Aron Heintz, Joon Suh Choi, Jordan Batchelor, Mehrnoush Karimi, and Agnes Malatinszky. 2023. A large-scaled corpus for assessing text readability. *Behavior Research Methods* 55, 2 (2023), 491–507.

[10] Scott A Crossley, Stephen Skalicky, and Mihai Dascalu. 2019. Moving beyond classic readability formulas: New methods and new models. *Journal of Research in Reading* 42, 3-4 (2019), 541–561.

[11] Edgar Dale and Jeanne S Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin* (1948), 37–54.

[12] Edgar Dale and Jeanne S Chall. 1949. The concept of readability. *Elementary English* 26, 1 (1949), 19–26.

[13] Elnaz Davoodi and Leila Kosseim. 2016. On the Contribution of Discourse Structure on Text Complexity Assessment. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 166–174.

[14] Diana DeStefano and Jo-Anne LeFevre. 2007. Cognitive load in hypertext reading: A review. *Computers in human behavior* 23, 3 (2007), 1616–1641.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 4171–4186.

[16] Sidney D'Mello, Andrew Olney, Claire Williams, and Patrick Hays. 2012. Gaze tutor: A gaze-reactive intelligent tutoring system. *International Journal of human-computer studies* 70, 5 (2012), 377–398.

[17] Benjamin D Douglas, Patrick J Ewell, and Markus Brauer. 2023. Data quality in online human-subjects research: Comparisons between MTurk, Prolific, CloudResearch, Qualtrics, and SONA. *Plos one* 18, 3 (2023), e0279720.

[18] Budmonde Duinkharjav, Kenneth Chen, Abhishek Tyagi, Jiayi He, Yuhao Zhu, and Qi Sun. 2022. Color-perception-guided display power reduction for virtual reality. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–16.

[19] Mary C Dyson and Mark Haselgrove. 2001. The influence of reading speed and line length on the effectiveness of reading from screen. *International Journal of Human-Computer Studies* 54, 4 (2001), 585–612.

[20] Lisa Jo Elliott, Medina Ljubijanac, and Danielle Wieczorek. 2020. The effect of screen size on reading speed: A comparison of three screens to print. In *Advances in Human Factors in Training, Education, and Learning Sciences: Proceedings of the AHFE 2019 International Conference on Human Factors in Training, Education, and Learning Sciences, July 24-28, 2019, Washington DC, USA 10*. Springer, 103–109.

[21] Andreas Fink and Aljoscha C Neubauer. 2001. Speed of information processing, psychometric intelligence: And time estimation as an index of cognitive load.

Personality and Individual Differences 30, 6 (2001), 1009–1021.

[22] Thomas François and Eleni Miltsakaki. 2012. Do NLP and machine learning improve traditional readability formulas?. In *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations*. 49–57.

[23] Gregor Franken, Anja Podlesek, and Klementina Mozina. 2015. Eye-tracking study of reading speed from LCD displays: influence of type style and type size. *Journal of Eye Movement Research* 8, 1 (2015).

[24] Arthur C Graesser, Shulan Lu, George Tanner Jackson, Heather Hite Mitchell, Mathew Ventura, Andrew Olney, and Max M Louwerse. 2004. AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, & Computers* 36 (2004), 180–192.

[25] Arthur C Graesser, Danielle S McNamara, Max M Louwerse, and Zhiqiang Cai. 2004. Coh-Metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers* 36, 2 (2004), 193–202.

[26] Jesse W Grootjen, Philipp Thalhammer, and Thomas Kosch. 2024. Your Eyes on Speed: Using Pupil Dilation to Adaptively Select Speed-Reading Parameters in Virtual Reality. *Proceedings of the ACM on Human-Computer Interaction* 8, MHCI (2024), 1–17.

[27] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594* (2024).

[28] Robert Gunning. 1952. The technique of clear writing. *(No Title)* (1952).

[29] Duncan Haywood. 2024. Gpt-4o tokens per second comparable to gpt-3.5-turbo. Data and analysis. https://community.openai.com/t/gpt-4o-tokens-per-second-comparable-to-gpt-3-5-turbo-data-and-analysis/768559 Accessed: April 7, 2025.

[30] Nora Hollenstein, Jonathan Rotsztejn, Marius Troendle, Andreas Pedroni, Ce Zhang, and Nicolas Langer. 2018. ZuCo, a simultaneous EEG and eye-tracking resource for natural sentence reading. *Scientific data* 5, 1 (2018), 1–13.

[31] Te-Yuan Huang, Ramesh Johari, and Nick McKeown. 2013. Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming. In *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*. 9–14.

[32] Stephen Hutt, Joseph F Grafsgaard, and Sidney K D'Mello. 2019. Time to scale: Generalizable affect detection for tens of thousands of students across an entire school year. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–14.

[33] Stephen Hutt, Kristina Krasich, James R. Brockmole, and Sidney K. D'Mello. 2021. Breaking out of the lab: Mitigating mind wandering with gaze-based attention-aware technology in classrooms. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[34] Matthew Jörke, Shardul Sapkota, Lyndsea Warkenthien, Niklas Vainio, Paul Schmiedmayer, Emma Brunskill, and James A Landay. 2025. GPTCoach: Towards LLM-Based Physical Activity Coaching. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–46.

[35] Antony William Joseph and Ramaswamy Murugesh. 2020. Potential eye tracking metrics and indicators to measure cognitive load in human-computer interaction research. *J. Sci. Res* 64, 1 (2020), 168–175.

[36] Daniel Jurafsky and James H. Martin. 2025. Discourse Coherence. In *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* (3rd ed.). Chapter 24. https://web.stanford.edu/~jurafsky/slp3/24.pdf Online manuscript released January 12, 2025.

[37] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Sotirios Xydis, and Dimitrios Soudris. 2024. Slo-aware gpu dvfs for energy-efficient llm inference serving. *IEEE Computer Architecture Letters* 23, 2 (2024), 150–153.

[38] J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. (1975).

[39] Thomas Kosch, Jakob Karolus, Johannes Zagermann, Harald Reiterer, Albrecht Schmidt, and Paweł W Woźniak. 2023. A survey on measuring cognitive workload in human-computer interaction. *Comput. Surveys* 55, 13s (2023), 1–39.

[40] Thomas Kosch, Albrecht Schmidt, Simon Thanheiser, and Lewis L Chuang. 2020. One does not simply RSVP: mental workload to select speed reading parameters using electroencephalography. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[41] Naveen Kumar and Jyoti Kumar. 2016. Measurement of cognitive load in HCI systems using EEG power spectrum: an experimental study. *Procedia Computer Science* 84 (2016), 70–78.

[42] Sri Hastuti Kurniawan and Panayiotis Zaphiris. 2001. Reading online or on paper: which is faster? (2001).

[43] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

[44] Sébastien Lallé, Dereck Toker, and Cristina Conati. 2019. Gaze-driven adaptive interventions for magazine-style narrative visualizations. *IEEE Transactions on Visualization and Computer Graphics* 27, 6 (2019), 2941–2952.

[45] Sébastien Lallé, Dereck Toker, Cristina Conati, and Giuseppe Carenini. 2015. Prediction of users' learning curves for adaptation while using an information visualization. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*. 357–368.

[46] Christian Lander, Marco Speicher, Denise Paradowski, Norine Coenen, Sebastian Biewer, and Antonio Krüger. 2015. Collaborative newspaper: Exploring an adaptive scrolling algorithm in a multi-user reading scenario. In *Proceedings of the 4th International Symposium on Pervasive Displays*. 163–169.

[47] Etienne Le Sueur and Gernot Heiser. 2010. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 international conference on Power aware computing and systems*. 1–8.

[48] Hanchen Li, Yuhan Liu, Yihua Cheng, Siddhant Ray, Kuntai Du, and Junchen Jiang. 2024. Eloquent: A More Robust Transmission Scheme for LLM Token Streaming. In *Proceedings of the 2024 SIGCOMM Workshop on Networks for AI Computing*. 34–40.

[49] Zikun Li, Zhuofu Chen, Remi Delacourt, Gabriele Oliaro, Zeyu Wang, Qinghan Chen, Shuhuai Lin, April Yang, Zhihao Zhang, Zhuoming Chen, et al. 2025. AdaServe: SLO-Customized LLM Serving with Fine-Grained Speculative Decoding. *arXiv preprint arXiv:2501.12162* (2025).

[50] Neiwen Ling, Guojun Chen, and Lin Zhong. 2024. TimelyLLM: Segmented LLM Serving System for Time-sensitive Robotic Applications. *arXiv preprint arXiv:2412.18695* (2024).

[51] Fengkai Liu, Tan Jin, and John SY Lee. 2025. Automatic readability assessment for sentences: neural, hybrid and large language models. *Language Resources and Evaluation* (2025), 1–32.

[52] Jiachen Liu, Jae-Won Chung, Zhiyu Wu, Fan Lai, Myungjin Lee, and Mosharaf Chowdhury. 2024. Andes: Defining and enhancing quality-of-experience in llm-based text streaming services. *arXiv preprint arXiv:2404.16283* (2024).

[53] Paul Joe Maliakel, Shashikant Ilager, and Ivona Brandic. 2025. Investigating Energy Efficiency and Performance Trade-offs in LLM Inference Across Tasks and DVFS Settings. *arXiv preprint arXiv:2501.08219* (2025).

[54] Lena Mamykina, Elizabeth Mynatt, and Michael A Terry. 2001. Time aura: Interfaces for pacing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 144–151.

[55] Emma Marsden, Sophie Thompson, and Luke Plonsky. 2018. A methodological synthesis of self-paced reading in second language research. *Applied Psycholinguistics* 39, 5 (2018), 861–904.

[56] Xinxin Mei, Qiang Wang, and Xiaowen Chu. 2017. A survey and measurement study of GPU DVFS on energy conservation. *Digital Communications and Networks* 3, 2 (2017), 89–100.

[57] Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K Dey, and Hideyuki Tokuda. 2015. Attelia: Reducing user's cognitive load due to interruptive notifications on smart phones. In *2015 IEEE international conference on pervasive computing and communications (PerCom)*. IEEE, 96–104.

[58] OpenAI. 2023. What are tokens and how to count them. https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them Accessed: April 7, 2025.

[59] Gustav Öquist and Mikael Goldstein. 2001. Adaptive rapid serial visual presentation. *Unpublished Thesis, Uppsala University, Uppsala, Sweden* (2001).

[60] Gustav Öquist and Mikael Goldstein. 2003. Towards an improved readability on mobile devices: evaluating adaptive rapid serial visual presentation. *Interacting with Computers* 15, 4 (2003), 539–558.

[61] Gustav Öquist and Kristin Lundin. 2007. Eye movement study of reading text on a mobile phone using paging, scrolling, leading, and RSVP. In *Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*. 176–183.

[62] Nirmal Patel, Pooja Nagpal, Tirth Shah, Aditya Sharma, Shrey Malvi, and Derek Lomas. 2023. Improving mathematics assessment readability: Do large language models help? *Journal of Computer Assisted Learning* 39, 3 (2023), 804–822.

[63] Ildikó Pilán, Elena Volodina, and Richard Johansson. 2014. Rule-based and machine learning approaches for second language sentence-level readability. In *Proceedings of the ninth workshop on innovative use of NLP for building educational applications*. 174–184.

[64] Jan L Plass, Roxana Moreno, and Roland Brünken. 2010. Cognitive load theory. (2010).

[65] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[66] Pragnya Ramjee, Mehak Chhokar, Bhuvan Sachdeva, Mahendra Meena, Hamid Abdullah, Aditya Vashistha, Ruchit Nagar, and Mohit Jain. 2025. ASHABot: an LLM-powered chatbot to support the informational needs of community health workers. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–22.

[67] Keith Rayner, Elizabeth R Schotter, Michael EJ Masson, Mary C Potter, and Rebecca Treiman. 2016. So much to read, so little time: How do we read, and can speed reading help? *Psychological Science in the Public Interest* 17, 1 (2016), 4–34.

[68] Reza Rejaie, Mark Handley, and Deborah Estrin. 2000. Layered quality adaptation for Internet video streaming. *IEEE Journal on Selected Areas in Communications* 18, 12 (2000), 2530–2543.

[69] Antonios Saravanos, Stavros Zervoudakis, Dongnanzi Zheng, Neil Stott, Bohdan Hawryluk, and Donatella Delfino. 2021. The hidden cost of using Amazon Mechanical Turk for research. In *HCI International 2021-Late Breaking Papers: Design and User Experience: 23rd HCI International Conference, HCII 2021, Virtual Event, July 24–29, 2021, Proceedings 23*. Springer, 147–164.

[70] Erik Schils and Pieter de Haan. 1993. Characteristics of sentence length in running text. *Literary and linguistic computing* 8, 1 (1993), 20–26.

[71] Elliot Schumacher, Maxine Eskenazi, Gwen Frishkoff, and Kevyn Collins-Thompson. 2016. Predicting the Relative Difficulty of Single Sentences With and Without Surrounding Context. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1871–1881.

[72] Sally E Shaywitz and Bennett A Shaywitz. 2005. Dyslexia (specific reading disability). *Biological psychiatry* 57, 11 (2005), 1301–1309.

[73] Eva Siegenthaler, Laura Schmid, Michael Wyss, and Pascal Wurtz. 2012. LCD vs. E-ink: An Analysis of the Reading Behavior. *Journal of Eye Movement Research* 5, 3 (2012).

[74] Shubham Singh. 2025. ChatGPT Statistics (March 2025): Number of Users & Queries. *DemandSage* (28 February 2025). https://www.demandsage.com/chatgpt-statistics/

[75] Soroosh Solhjoo, Mark C Haigney, Elexis McBee, Jeroen JG van Merrienboer, Lambert Schuwirth, Anthony R Artino Jr, Alexis Battista, Temple A Ratcliffe, Howard D Lee, and Steven J Durning. 2019. Heart rate and heart rate variability correlate with clinical reasoning performance and self-reported measures of cognitive load. *Scientific reports* 9, 1 (2019), 14668.

[76] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.

[77] Zhenheng Tang, Yuxin Wang, Qiang Wang, and Xiaowen Chu. 2019. The impact of GPU DVFS on the energy and performance of deep learning: An empirical study. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*. 315–325.

[78] MiM Taylor, C Douglas Creelman, et al. 1967. PEST: Efficient estimates on probability functions. *Journal of the acoustical society of America* 41, 4 (1967), 782–787.

[79] TOI Tech Desk. 2025. ChatGPT down: AI chatbot back after global outage. *The Times of India* (30 March 2025). https://timesofindia.indiatimes.com/technology/tech-news/chatgpt-down-ai-chatbot-outage-affecting-ghibli-style-photo-generation-globally/articleshow/119754435.cms

[80] Sowmya Vajjala and Ivana Lučić. 2018. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*. 297–304.

[81] Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the seventh workshop on building educational applications using NLP*. 163–173.

[82] VentureBeat. 2024. How enterprises are using open source LLMs: 16 examples. *VentureBeat* (30 Jan. 2024). https://venturebeat.com/ai/how-enterprises-are-using-open-source-llms-16-examples Accessed July 2025.

[83] Shaun Wallace, Zoya Bylinskii, Jonathan Dobres, Bernard Kerr, Sam Berlow, Rick Treitman, Nirmal Kumawat, Kathleen Arpin, Dave B Miller, Jeff Huang, et al. 2022. Towards individuated reading experiences: Different fonts increase reading speed for different individuals. *ACM Transactions on Computer-Human Interaction (TOCHI)* 29, 4 (2022), 1–56.

[84] Chen Wu. 2025. Test Time Compute: Balancing Throughput, Speed, and Quality in LLM Thinking. https://medium.com/@chenwuperth/test-time-compute-balancing-throughput-speed-and-quality-in-llm-thinking-8428c5cf757c. *Medium* (26 Mar 2025).

[85] Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics* 3 (2015), 283–297.

[86] Peifeng Yin, Ping Luo, Wang-Chien Lee, and Min Wang. 2013. Silence is also evidence: interpreting dwell time for recommendation from psychological perspective. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 989–997.

[87] Johannes Zagermann, Ulrike Pfeil, and Harald Reiterer. 2016. Measuring cognitive load using eye tracking technology in visual computing. In *Proceedings of the sixth workshop on beyond time and errors on novel evaluation methods for visualization*. 78–85.

[88] Xi Zheng, Zhuoyang Li, Xinning Gui, and Yuhan Luo. 2025. Customizing emotional support: How do individuals construct and interact with LLM-powered chatbots. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–20.

[89] Yihao Zhu, Zhoutong Ye, Yichen Yuan, Wenxuan Tang, Chun Yu, and Yuanchun Shi. 2025. AutoPBL: An LLM-powered Platform to Guide and Support Individual Learners Through Self Project-based Learning. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–26.