COMPUTER SCIENCE

Peace Space: An Offline, Context-Aware Vocabulary and Idiom Support Architecture for Mobile Digital Reading

Arnav Chaudhary[1], Ayush Kumar Jha[1], S. Charishma[1], Abhinandana A. Poojary[1], Sunil Maggu[2]

[1]Department of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India
[2]Mentor, Maharaja Agrasen Institute of Technology, Delhi, India

Published November 2026

Abstract

The act of digitally reading in English places a high degree of extraneous load upon a reader who is not native to the language, particularly where unknown or technical words, phrases and set expressions arise. The existing process involves highlighting a passage, navigating away from the document, performing a search (e.g. via a dedicated dictionary app or internet), identifying the relevant definition(s) and re-navigating back. Cognitive load literature shows that these processes cause loss of short-lived memory traces and degrade the reader's ability to build up an integrated understanding of the content. In this paper we describe the high-level design and implementation of 'Peace Space', a standalone Android application which aims to deliver contextual definitions of individual words and phrases directly within the document at the point of query. We separate the functionality currently achieved, i.e. extracting a layer of the document content and running an SLM on the device, from future features we intend to incorporate. Our primary addition here is a 'Tiered Early-Exit' cascade of operations that can efficiently return clear-cut answers using a pre-populated static word-store, before deferring to (quantised) SLM computation for more challenging cases. Finally, we discuss the limitations of the Android OS in relation to displaying content over other apps and receiving highlight actions, and outline a test framework to support our claims.

Keywords: Android app design; cognitive load; idiom; on-device; digital reading; NLP at the edge; small models; WSD

1 Introduction

As more people turn to their smartphones as a means to read English texts for study and work, there has been significant change in this space. A major bottleneck arises for learners of English, who may struggle with specialized terminology, words with multiple meanings, and fixed expressions. In practice, using their smartphone to find out the meaning of such a word will typically involve: selecting the target, copying to the device clipboard, switching to another applications like dictionary, Google search or something chat bot, opening a reference source, searching its contents for the relevant definition (often without benefit of surrounding context, as one word might hold different meanings based on the context and state and non native speakers and readers often struggle to fit the meaning of found word into that given context by themselves), and returning to the source.

Cognitive Load Theory suggests that humans have very low bandwidth in their working memory, such that any additional load not directly contributing to the formation of schemas will negatively affect understanding [1]. Research into tasks imposed on readers has shown these effects most strongly for high-level, synthesizing forms of reading, i.e. identifying overarching ideas across

multiple consecutive sentences, although simple fact retrieval was largely unaffected [2]. Smartphones also present challenges of unknown words to English-language readers, and the need to provide tailored support to each reader can interfere with their normal pattern of reading [3].

With significant lookup costs, readers will experience many interruptions to their focus, or simply ignore unknown vocabulary and expressions, both of which reduce reading fluency, and the complete understanding of the text.

Current solutions can only partially solve this need. Offline wordbooks provide fast lookups, but give no further help with words that may have multiple senses depending on their usage, or for idiomatically-charged terms. LLMs hosted online are able to produce helpful, context-sensitive definitions; however they incur a delay due to communication over the internet, depend upon an always-on connection, and force the user to send their (possibly private) reading content to a third party.

In response, we propose and describe Peace Space, an Android-based local reading aid which can provide contextual definitions and explanations of key terms and idioms directly within the pdf document, with no need to leave the app or rely on third party services. As this is still a work in progress, we distinguish between functionality currently realized and further development required. The main outputs of our study are as follows:

1. A fully local Android platform for reading documents and identifying terms of interest (and associated context), ready to prompt LLM-generated definition(s).
2. **A two-tier early-exit cascade:** In order to operate within the constraints of mobile device memory and compute time but still accurately resolve common English idiom and multiple-meaning cases, our solution splits work between a pre-populated static dictionary for simple 1-word queries and a quantized SLM model where additional context-aware reasoning is needed.
3. **Platform-Aware Design and Evaluation Protocol:** We analyze critical Android security and intent boundaries including ACTION_PROCESS_TEXT context restrictions and TYPE_APPLICATION_OVERLAY constraints and propose a consistent set of hardware, NLP and user study conditions for assessing model performance.

2 Related work

2.1 Cognitive load and augmented reading interfaces

Sweller [1] showed that reducing the amount of extraneous load imposed on an individual allows them to maintain sufficient cognitive capacity when processing dense material. Applying this finding to e-reading, Foroughi et al. [2] carried out three experiments showing that even short periods of performing a concurrent task degraded the temporary 'trace' of text held in working memory (WM), reducing readers' ability to answer higher-order concept questions. Studies of m-reading similarly found that short periods of distraction impaired the temporary representation of text in WM, decreasing the ability to answer higher-order questions requiring the integration of multiple ideas.

Research on mobile learning for learners of English as a second language has also highlighted the importance of developing tailored m-learning approaches.

A key motivation behind many augmented reading approaches in HCI has been to remove the need for users to leave their current document to look up term meanings. ScholarPhi [4], proposed by Head et al., presents a form of augmented reading that overlays term-specific definitions onto a user's PDF, at their relevant location in the page. Their system provides these definitions on-demand for

scientific terminology and mathematical expressions. Although it showed promise with respect to improving the experience of scientists, its design required significant backend computation specific to science and was implemented for desktop. In contrast, we focus on enabling augmented reading for low-power Android smartphones, ensure complete local operation, and apply in-document augmentation to a broader range of words and phrases likely to challenge ESL users.

2.2 Word sense disambiguation and idiomatic resolution

Ascertaining the target sense of a given word in context has been addressed within NLP as the task of Word Sense Disambiguation (WSD) [5], with prior work dividing these methods into 3 broad classes:

- **Knowledge-Based Methods:** These techniques leverage structured lexical inventories such as WordNet or BabelNet [5], [7].

One class of approaches makes use of dictionary overlaps (e.g. using Lesk-type measures) to compute a match between each candidate's definition and its immediate context [6]. While broadly applicable and low-cost, these methods typically fail to discriminate closely related senses or idiomatically used expressions with little shared surface form.

- **Supervised ML:** Models can be learnt for individual word senses using, for example, the SemCor corpus [5]. SVMs were long considered state of the art for this type of problem in some WSD tasks.

For example, Chen et al. [8] used an active-learning approach to train SVMs to obtain good disambiguation performance on biomedical abstracts, with reduced labeling effort. A significant challenge with these approaches is obtaining sufficient training data and their limited ability to cope with novel domain-specific terms and fixed multi-word expressions.

- **Generative LMs:** Recent advances in large language models (LLMs), and more recently small language models (SLMs) that can be deployed on end devices, have demonstrated the capacity to disambiguate and explain idioms via free-form generation, as discussed in Navigli's survey [9].

Although generative models are effective at fluent, surface-based sense disambiguation, they tend to underperform on infrequent and less dominant word senses. Their performance could be improved through sense-focused metrics and complementary approaches. On-device deployment also requires control of model size, time-to-token, and device heating.

3 System architecture and implementation status

To maintain engineering transparency, we divide the discussion of Peace Space into three levels: existing repositories, further algorithmic improvements, and future work.

3.1 Currently implemented core architecture

The current prototype operates as an offline Android reading application comprising of two functional components:

1. **Document Rendering and Context Extraction:** Digital documents are rendered so that their textual content can be shown. Users can select a single word or a sequence of consecutive words (e.g. idiom) using a touchscreen interface.

When a word or phrase is selected then it will trigger an extraction function to retrieve the chosen span w and its surrounding local context C. This consists of 1 preceding and following sentence, i.e. the full containing sentence plus up to 1 sentence before and after.

2. **On-Device Language Inference Pipeline:** It is then converted to a short instruction and run on-device via a SLM: Qwen2.5-0.5B-Instruct Q4_K_M running on llama.cpp, constrained to produce at most 32 tokens. The response will be generated as a short relevant message within the in-situ sheet avoiding any transmission of the user's input.

3.2 Proposed plan: tiered early-exit cascade

Because autoregressive SLMs will require one token at a time applying them to each selected word will lead to huge delays and power consumption.

Alternatively, if we run an offline classical model (e.g. SVM) before invoking the SLM for each input we still bear the cost of autoregressively generating tokens as well as being limited to training on a per-lemma basis.

We therefore redesign our architecture to use a tiered early-exit with a simple classifier and 2 branches:

- **Tier 1 (Offline Fast Path):** The deterministic router will check the selected term w. For cases where w corresponds to one entries in our pre-created local SQL DB of words (from WordNet, Wiktionary etc) we can return a short meaning to the user in <~0.1s and leave the SLM unused.
- **Tier 2 (Contextual SLM Slow Path):** When w is identified as a potential multi-word selection, OOV technical vocabulary the request (w, C) is routed to the quantised on-device SLM. By progressively returning tokens we will be able to deliver context as soon as TTFT.

This helps in reducing the user experience of waiting for the model's response.

3.3 Ongoing extension: cross-application overlay and Android constraints

When trying to apply in-situ lookup to external PDF applications and browser we face various Android OS-level restrictions which leads to the architecture of our solution:

- **Intent availability:** The default ACTION_PROCESS_TEXT intent registers an action in the system text-selection toolbar and passes only the explicitly highlighted string via EXTRA_PROCESS_TEXT; it does not return adjacent paragraph text from third-party apps.

As ACTION_PROCESS_TEXT is bound to an Activity (as opposed to a hidden Service), some form of low-profile 'trampoline' Activity is required. This would be an Activity which consumes the Intent and starts the Overlay Service then exits.

- **Overlay rendering and touch passthrough:** An in-situ definitions overlay requires TYPE_APPLICATION_OVERLAY and the SYSTEM_ALERT_WINDOW flag to be granted by the user, which may not be available on low-memory Android Go devices. Scrolling beneath the overlay can be enabled by setting both FLAG_NOT_FOCUSABLE and FLAG_NOT_TOUCH_MODAL, while respecting Android 12+ untrusted-touch security limits (which block touch passthrough above a default overlay opacity of 0.8).

Privacy and policy considerations mean that we do not scrape external apps using the Android Accessibility Service or MediaProjection API. Therefore, all sentence context is parsed within the app's built-in Peace Space reader. The ACTION_PROCESS_TEXT screen overlay.

4 Architectural comparison and evaluation

Table I compares Peace Space against various mobile PDF and e-book reading tools across privacy, idiom meaning, contextual disambiguation and licensing. Various offline readers such as KOReader and Moon+ Reader preserve privacy through local dictionaries but return context-blind definitions. On the other hand cloud-assisted readers such as Adobe Acrobat AI Assistant require continuous connectivity and paid subscriptions, while emerging local-AI readers (Shelfy, re/read) lack published sense-disambiguation benchmarks or mobile thermal budgets.

Table 1. Architectural comparison of mobile reading and lookup approaches.

System / Approach	Execution & Privacy	Phrase & Context Resolution	Licensing & Cost	Key Operational Trade-Off
Adobe Acrobat Reader (Liquid Mode & AI Assistant) [3]	Cloud servers; internet required for AI features	Multi-PDF Q&A, summaries, citations, and OCR	Free viewer; paid DC subscription / AI add-on	Unavailable offline for AI Q&A; limited local network storage integration
Amazon Kindle / Word Wise (+ WordDumb)	Offline built-in dictionaries and pre-built sidecars	Single-tap definitions and static inline glosses	Proprietary ecosystem; open-source Calibre plugin	Requires pre-generated sidecar files; cannot disambiguate arbitrary user PDFs
KOReader + StarDict (Wiktionary / WikDict) [1]	100% offline local StarDict index (.idx, .dict.dz)	Word and drag-selected phrase lookup across PDF/EPUB	Open-source reader and free dictionary packs	Context-blind lookup returns unranked senses and misses non-compositional idioms
Moon+ Reader + External Apps (GoldenDict / Look Up) [2]	Offline external dictionary invocation via Android intents	Merges WordNet 3.0 / Webster's; handles 1–3 word spans	Proprietary app (free/Pro) + third-party apps	App switching and unranked entries increase visual interruption during reading [5]
Shelfy / re/read (Local-AI Readers) [4]	100% offline on-device AI or Bergamot + StarDict	Difficulty-graded inline glosses and flashcard export	Free Android app / AGPL-3.0 browser extension	No published WSD/ idiom accuracy scores or mobile thermal soak telemetry
Peace Space (Proposed)	100% offline local execution; zero server upload	2-tier early-exit cascade (Fast-Path + on-device SLM)	Local open-weights / dictionary stack	Bounds memory and thermal load while minimizing reading interruption

To validate Peace Space rigorously without relying on unverified empirical claims, we establish a three-part evaluation framework combining analytical cascade modeling, standardized NLP benchmark targets, and a planned human comprehension trial.

4.1 On-device latency and resource profiling

End-to-end lookup latency is defined as:

$$T_{\text{total}} = T_{\text{extract}} + T_{\text{route}} + T_{\text{infer}} + T_{\text{render}} \quad (1)$$

where T_{extract} is PDF text and context-window extraction time, T_{route} is routing cost, T_{infer} is either Fast-Path dictionary lookup or Slow-Path small language model (SLM) time, and T_{render} is UI bottom-sheet

rendering time. Because ambiguous queries adds routing overhead onto SLM generation expected user latency follows the model:

$$E[T] = pT_{\text{fast}} + (1 - p)T_{\text{slow}} \quad (2)$$

where p is the early-exit rate, T_{fast} the Fast-Path time and T_{slow} the Slow-Path time.

Cross-tier profiling spans mid-range and flagship Android hardware classes: mid-tier targets correspond to Nothing CE standard mid-range devices -Phone (2a) Plus (Dimensity 7350 Pro, 12 GB RAM), Phone (3a) (Snapdragon 7s Gen 3, 8/12 GB RAM), and CMF Phone 1 (Dimensity 7300, 6/8 GB RAM), while flagship targets correspond to the Samsung Galaxy S24 (Snapdragon 8 Gen 3 / Exynos 2400, 8 GB RAM) and Galaxy S25 (Snapdragon 8 Elite, 12 GB RAM). Under the profiling harness, 120 fixed prompts (60 single words and 60 idioms) are instrumented via SystemClock.elapsedRealtimeNanos() and adb logcat, peak resident memory (VmHWM and Total PSS) is recorded via adb shell dumpsys meminfo -s, and thermal stability is evaluated across 15-minute warm soak sessions using PowerManager.getThermalHeadroom(), PerformanceHintManager and adb shell dumpsys thermalservice [7].

Table II summarizes published reference engine benchmarks alongside Peace Space's analytical latency and resource targets. Modeling a 25 ms Fast Path (range 16–47 ms: 5–15 ms extraction, 1–5 ms routing, 2–10 ms lookup, 8–17 ms UI render) and concise 10–15 token (3–5 word [8]) Slow-Path totals of 700 ms (flagship) and 1,600 ms (mid-tier) yields expected cascade latencies of 227.5 ms (flagship) and 497.5 ms (mid-tier) at a 70% early-exit rate (ranging from 160.0–295.0 ms and 340.0–655.0 ms across 80%–60% exit rates).

Table 2. Published reference baselines and analytical target endpoints.

Metric	Published Reference Baseline	Flagship Target (Galaxy S24 / S25)	Mid-Tier Target (Nothing 3a / 2a Plus)	Telemetry Interface
PDF Load & Span Extraction	MuPDF Core ~2 MB (0.1 s/doc desktop avg); Nutrient Android SDK 120 ms load [9]	Cold open <= 150 ms; span extract 5–15 ms	Cold open <= 220 ms; span extract 8–20 ms	elapsedRealtimeNanos()
Tier-1 Route & Fast Path	MediaPipe MobileBERT 57.68 ms CPU (0.14 ms embedding) on Pixel 6 [10]; sparse routing < 100 ms	Early-exit >= 70%; Fast Path 16–47 ms (25 ms median)	Early-exit >= 70%; Fast Path 20–55 ms (<= 40 ms median)	Per-stage adb logcat
Slow-Path SLM Decode & JSON Validity	Gemma 2B INT4: 22–23 tok/s (flagship) vs. 6–8 tok/s (SD 7s Gen 2); Llama 3.2 1B: 10–14 tok/s (SD 7s Gen 2), 11.18 tok/s 2-core pinned (Dimensity 7025) [11], [12]; grammar JSON ~99% at 390 ms [13]	>= 20 tok/s; TTFT <= 300 ms; >= 98% valid schema	>= 10 tok/s (<= 1.7B INT4, 2-core pinned); TTFT <= 600 ms; >= 95% schema validity	Token timestamp diff
Blended Cascade Latency (Eq. 2)	Analytical model (25 ms Fast Path; 700 ms flagship / 1,600 ms mid-tier Slow Path)	227.5 ms at 70% exit (160.0 ms at 80%; 295.0 ms at 60%)	497.5 ms at 70% exit (340.0 ms at 80%; 655.0 ms at 60%)	Tap-to-render timer

Metric	Published Reference Baseline	Flagship Target (Galaxy S24 / S25)	Mid-Tier Target (Nothing 3a / 2a Plus)	Telemetry Interface
Peak Process Memory	Quantized 1B–2B SLMs: 1.1–1.9 GB footprint (VmHWM ~1.76 GB for Llama 3.2 1B Q4_0) [12]	Total PSS / VmHWM <= 2.2 GB	Total PSS / VmHWM <= 1.8 GB (Fast-Path fallback on < 4 GB RAM)	dumpsys meminfo -s
15-Min Thermal Soak & UI Frame Budget	Naive SD 8 Gen 3 drops 69% at 30 min (12.4 to 3.8 tok/s) vs. 77% retention via ADPF [7]; 120 Hz budget < 8.3 ms	>= 75% adapted peak tok/s at min 15; UI jank rise <= 5%	>= 70% adapted peak tok/s at min 15 via thermal headroom	getThermalHeadroom()

4.2 Disambiguation and idiom accuracy

Context resolution and explanation accuracy are evaluated across three configurations: (1) a static dictionary baseline, (2) the Fast-Path module alone, and (3) the complete Tiered Early-Exit Cascade. Single word is benchmarked under the unified WordNet 3.0 framework of Raganato trained on SemCor (226,036 annotations) and evaluated across Senseval-2, Senseval-3, SemEval-07, SemEval-13, SemEval-15 and their 7,253-item union (ALL). Idiom detection and interpretation are evaluated across MAGPIE (4,840 random test instances, paired with contrastive DICE items to mitigate annotation noise) SemEval-2022 Task 2a (916 English items), FLUTE (250 unseen-idiom NLI pairs), PIE-English (20,174 samples across 10 figurative classes) and a curated set of 50–100 naturally occurring idioms from scientific and journalistic articles.

Published evaluations show that general-purpose LLMs can trail compact fine-tuned encoders on idiomaticity (GPT-3.5-Turbo scores 0.645 macro-F1 on SemEval-2022 Task 2a zero-shot versus 0.707 for mBERT [17]), motivating Peace Space's compact Tier-1 router ahead of generative explanation. Table III lists verified literature baselines alongside Peace Space's accuracy targets.

Table 3. Published NLP baselines and Peace Space target accuracy endpoints.

Benchmark & Task	Static / Classical Baseline	Compact Encoder / Fast-Path Baseline	Supervised Ceiling	Peace Space Cascade Target
Raganato et al. WSD ALL (7,253 items) [14]	Lesk_ext = 48.6 F1; SemCor MFS = 64.8 F1; WN 1st Sense = 65.2 F1	Lesk_ext+emb = 63.7 F1; IMS = 68.4 F1; IMS-s+emb = 69.6 F1	~80 F1 (SemCor upper bound)	>= 75.0 F1 on ALL
MAGPIE Idioms (4,840 test items) [15]	Uncontextualized dictionary lookup	Context-compatibility neural encoder	0.955 macro-F1 (fine-tuned)	>= 0.90 macro-F1 (+ DICE check [16])
SemEval-2022 Task 2a (916 EN items) [17]	Static compound matching	mBERT: 0.707 zero-shot / 0.886 one-shot (GPT-3.5 = 0.645)	0.902 zero-shot / 0.964 one-shot	>= 0.75 zero-shot; >= 0.90 one-shot
FLUTE Idiom Subset (250 NLI pairs) [18]	Literal paraphrase mismatch	T5 = 0.791 macro-F1 (79.2% acc.); GPT-3.5 = 0.820	0.952 macro-F1 (fine-tuned)	>= 0.82 macro-F1 on unseen idioms
PIE-English (20,174 samples, 10 classes) [19]	Multinomial NB = 74.7% acc. (0.66 F1)	Fine-tuned BERT = 93.4% acc. (0.948 weighted F1)	93.4% acc. (BERT)	>= 90% acc. / >= 0.91 weighted F1

4.3 Controlled reading comprehension user study

Following the experimental framework of Foroughi et al. [2], we plan to run a within-subjects counterbalanced experiment with 16-24 ESL undergraduates. Subjects will be presented with a pair of equivalent-length texts with target words and phrases they will either use an external dictionary or web-browser, or Peace Space. Our measures are: (1) composite score from concept-based Qs to test subject's overall understanding of the texts (2) time taken to complete each task (3) self reported mental workload (NASA-TLX) and (4) word and phrase recall immediately followed by reading.

5 Limitations and future work

Various technical and experimental limitations define the current scope of this work:

1. **Lack of experimental evidence:** We have so far collected no direct data on student reading ability. This will require testing as detailed in the section 'Architectural comparison and evaluation'.
2. **Reliant on digital, text-layered content:** As described above, our method relies on the presence of a digitized text layer within the document(s). To extend this to scanned material we would need to implement a low-overhead OCR process on the user's device. An OCR capability will be included in later versions.
3. **Multiple apps requiring permission to display a system-wide floating window:** These will require the SYSTEM_ALERT_WINDOW flag, which may be treated differently depending on OEM implementation of Android or use of Android Go.
4. **Small-Model Hallucinations:** Heavily-discrete <1B models may return incorrect definitions for niche words and some common idioms [9]. We plan to test the use of model confidence scores and presenting an alternative definition as required.

6 Conclusion

In this work we described the system-level design of Peace Space, an offline Android app to decrease the mental cost of word lookups during reading for ESL users. We combined a realized local document parser and model inference engine with our suggested Tiered Early-Exit Cascade to enable resolution of single words (polysemy) and fixed phrases. Next steps include finalizing the Cascade, NLP methods and running a targeted experiment to measure effects on reading experience and inferred understanding.

References

[1] J. Sweller, "Cognitive load during problem solving: Effects on learning," *Cognitive Science*, vol. 12, no. 2, pp. 257-285, Apr. 1988, doi:10.1207/s15516709cog1202_4.

[2] C. K. Foroughi, N. E. Werner, D. Barragán, and D. A. Boehm-Davis, "Interruptions disrupt reading comprehension," *Journal of Experimental Psychology: General*, vol. 144, no. 3, pp. 704-709, 2015, doi:10.1037/xge0000074.

[3] R. Higashimura, K. Watanabe, A. Vargo, M. Iwata, A. Dengel, and K. Kise, "Estimating unknown English words from user smartphone reading behaviors," *IEEE Access*, vol. 12, pp. 140223-140234, 2024, doi:10.1109/ACCESS.2024.3457510.

[4] A. Head, K. Lo, D. Kang, R. Fok, S. Skjonsberg, D. S. Weld, and M. A. Hearst, "Augmenting scientific papers with just-in-time, position-sensitive definitions of terms and symbols," in *Proc*. 2021 CHI Conf. Human Factors in Computing Systems (CHI '21), 2021, Art. no. 81, pp. 1-16, doi: 10.1145/3411764.3445648.

[5] R. Navigli, "Word sense disambiguation: A survey," *ACM Computing Surveys*, vol. 41, no. 2, Art. no. 10, pp. 1-69, Feb. 2009, doi:10.1145/1459352.1459355.

[6] E. Agirre and P. Edmonds, Eds., Word Sense Disambiguation: Algorithms and Applications, Text, Speech and Language Technology, vol. 33. Dordrecht, The Netherlands: Springer, 2006.

[7] M. Bevilacqua, T. Pasini, A. Raganato, and R. Navigli, "Recent trends in word sense disambiguation: A survey," in *Proc*. 30th Int. Joint Conf. Artif. Intell. (IJCAI-21), 2021, pp. 4330-4338, doi: 10.24963/ijcai.2021/593.

[8] Y. Chen, H. Cao, Q. Mei, K. Zheng, and H. Xu, "Applying active learning to supervised word sense disambiguation in MEDLINE," *Journal of the American Medical Informatics Association*, vol. 20, no. 5, pp. 1001-1006, 2013, doi: 10.1136/amiajnl-2012-001244.

[9] R. Navigli, "Is word sense disambiguation dead in the LLM era?" in *Proc*. AAAI Conf. Artif. Intell. (AAAI-26), vol. 40, no. 46, pp. 39753