STM JOURNALS

Emerging Trends in
Languages
https://journals.stmjournals.com/etl

ISSN: 3049-0693
Volume 3, Issue 2, 2026
July–December
DOI (Journal): 10.37591/ETL

Research ETL

Font-Based Text Steganography for Covert Communication in Devanagari Script: A Review (2020–2026)

Mohd Kamil[1]*, Hitesh Singh[2], Aditee Mattoo[3]

Abstract

Steganography hides the fact that a communication exists rather than merely safeguarding its content. Text is the most challenging of all steganographic media because it contains the least redundancy and is the most sensitive to changes. Devanagari script, with its conjunct consonants, ligatures, matras, and non-linear glyph construction, makes these difficulties even harder, rendering rendering-dependent and OCR- based steganographic methods largely ineffective. This paper presents a systematic review of text steganography, with particular focus on font- based approaches and their applicability to Devanagari script, covering the period 2020 to 2026. The review classifies methods into six broad categories linguistic, format-based, Unicode-based, OCR-based, glyph- perturbation, font-metadata, and LLM-generative and analyses them across capacity, imperceptibility, robustness, determinism, and script awareness. A key finding is that despite the widespread digital use of Indic languages, virtually no high-capacity text steganography system addresses the structural complexity of Devanagari. Font-internal encoding specifically the use of font metadata tables in TrueType/OpenType fonts emerges as the most promising approach for achieving deterministic, OCR-free, and script-aware covert communication in Devanagari. The review outlines open challenges, evaluates existing methods against a unified metrics framework, and proposes future research directions including multi-script font systems, GAN-based glyph generation, hybrid semantic-visual encoding, and standardised Indic benchmarks.

Keywords: Text steganography, font-based steganography, covert communication, Devanagari script, glyph perturbation, font meta- data, Indic script steganography, OCR-free encoding, linguistic steganography.

INTRODUCTION

The art of steganography, which involves concealing a message to hide its very existence, is as old as recorded communication. In ancient Greece, messages were tattooed on shaved heads and concealed once the hair had grown back. In the digital era, steganography embeds secret information within ordinary carriers such as images, audio, video, and text, so that a casual observer suspects no communication is taking place at all [1]. This property is what makes steganography fundamentally different from cryptography: encryption secures the content of a message but makes it visible that a secret communication is occurring, which itself can attract suspicion or legal scrutiny. Steganography, by contrast, aims for complete concealment of the communication channel [2].

Text-based steganography is generally

***Author for Correspondence**
Mohd Kamil
E-mail: mohdkamil439529@gmail.com

[1,2]Student, Department of Computer Science and Engineering, Noida Institute of Engineering and Technology, Greater Noida, Uttar Pradesh, India.
[3]Assistant Professor, Department of Computer Science and Engineering, Noida Institute of Engineering and Technology, Greater Noida, Uttar Pradesh, India.

Received Date: June 29, 2026
Accepted Date: July 09, 2026
Published Date: July 10, 2026

Citation: Mohd Kamil, Hitesh Singh, Aditee Mattoo. Font-Based Text Steganography for Covert Communication in Devanagari Script: A Review (2020–2026). Emerging Trends in Languages. 2026; 3(2): 23–33p.

considered the most challenging variant. Unlike image or audio carriers, which contain rich perceptual redundancy that can absorb minor changes without detection, text carries almost no such redundancy [3]. Every character is semantically load-bearing. A single extra space, a misplaced punctuation mark, or an imperceptibly altered character shape can be detected visually by a careful reader, structurally by a diff tool, or statistically by a steganalyser. Text is also extremely susceptible to transformation: reformat- ting, copy-pasting, OCR conversion, or even changing the font can destroy any steganographic channel embedded in the structure or appearance of the text [4].

These difficulties multiply when the target script is Devanagari the writing system for Hindi, Marathi, Sanskrit, Nepali, and over one hundred other languages spoken by more than 600 million people across South Asia. Devanagari is an abugida: consonants carry an inherent vowel, and vowels are represented by diacritical marks (matras) attached above, below, or around the base consonant. Conjunct consonants formed when a halant (virama) character suppresses the inherent vowel and fuses two consonants into a single glyph introduce non- linear visual compositions that render fundamentally differently from Latin text [5]. The Unicode block (U+0900–U+097F) contains more than 128 code points, and Unicode normalisation processes can silently collapse variant representations of the same character, destroying any channel that relies on code-point variation [6].

Font-based steganography where the font file rather than the rendered text serves as the covert channel offers an entirely different approach. By embedding hidden data in the internal structure of font files, such methods are independent of document layout, rendering engines, and OCR. Two principal families exist: glyph-perturbation methods, which alter the geometric shapes of glyphs in the font's outline tables, and font-metadata methods, which encode data in the font's name table or custom metadata records. The glyph-perturbation approach, exemplified by FontCode, has demonstrated high imperceptibility and print-scan robustness but was designed exclusively for Latin glyphs and uses a Latin-specific codebook. Font-metadata methods avoid glyph-level rendering dependencies entirely and are in principle script-agnostic, making them the most immediately applicable approach for Devanagari [7].

This paper provides a systematic review of text steganography techniques from 2020 to 2026, with focus on font-based approaches and their applicability to Devanagari script. The review aims to:

1. Classify and analyse existing approaches across a unified set of evaluation metrics.
2. Characterise the structural properties of Devanagari that make it uniquely challenging.
3. Identify the most promising technical directions for font-based Devanagari steganography.
4. Outline open research gaps and future directions.

The remainder of the paper is organised as follows. Section 2 provides background on steganographic principles and evaluation metrics. Section 3 reviews text steganography methods from 2020 to 2026 across all major categories. Section 4 analyses the specific challenges posed by Devanagari script. Section 5 examines font-based approaches in detail. Section 6 presents a comparative analysis. Section 7 identifies research gaps and future directions. Section 8 concludes the review.

BACKGROUND AND EVALUATION FRAMEWORK

Text steganography conceals secret information within ordinary text so that the existence of the hidden message remains undetectable. A typical steganographic system involves a sender (Alice), a receiver (Bob), and an adversary (Eve), where the sender embeds a secret message into a cover text to produce a stego text for secure communication.

Steganographic Principles

A steganographic system involves three parties: the sender (Alice), the receiver (Bob), and a potential passive or active adversary (Eve). Alice embeds a secret message M into a cover object C to produce a stego object S, which is transmitted to Bob through a channel that Eve can observe. The

fundamental security goal is that *S* must be statistically indistinguishable from *C* Eve must not be able to determine whether *S* contains hidden information [8].

Text steganography operates on cover text documents. The three primary requirements of any text steganography system are: imperceptibility (the stego text must be visually and linguistically indistinguishable from natural text), capacity (the system must embed a practically useful amount of hidden information), and robustness (the hidden data must survive the transformations the stego text is likely to undergo during transmission or storage). In practice these three properties are in tension: increasing capacity typically reduces imperceptibility or robustness, and vice versa [9].

Evaluation Metrics

The literature has converged on a set of standard metrics for evaluating text steganography systems, summarised in Table 1. Of particular importance for this review is the metric of script awareness the degree to which a method correctly handles the structural properties of its target script rather than assuming Latin-like character independence.

For Devanagari specifically, two additional dimensions apply: glyph segmentation accuracy (whether glyph boundaries in conjunct clusters are correctly identified) and normalisation stability (whether embedded data survives Unicode NFC/NFD normalisation applied by text editors and communication platforms).

REVIEW OF TEXT STEGANOGRAPHY METHODS (2020–2026)

Linguistic-based text steganography hides secret information by making semantic or syntactic modifications, such as synonym substitution or sentence restructuring, while preserving the natural meaning and readability of the text.

Linguistic-Based Methods

Linguistic steganography encodes data by making semantic or syntactic substitutions within natural language text replacing words with synonyms, reordering clauses, or paraphrasing sentences in ways that preserve meaning while encoding bits. Early work relied on manually crafted synonym dictionaries and context-free grammar (CFG) mimic functions. These produced natural-looking text but achieved only 0.5–2 bits per word and suffered from semantic drift.

Between 2020 and 2023, neural language models transformed this category. Yang et al. proposed VAE-Stega, a variational autoencoder that learns the latent semantic space of text and embeds data by controlled sampling, achieving approximately 3 bits per word with improved imperceptibility [10]. Ding et al. proposed Discop, which constructs distribution copies steganographic mappings that by design do not alter the token probability distribution of the underlying language model enabling provably secure embedding with state-of-the-art statistical imperceptibility [11].

The critical limitation of all linguistic methods is language-model dependency: sender and receiver must share identical models and vocabularies. No published linguistic steganography system addresses Devanagari text generation. Hindi's morphological richness, postpositional syntax, and case-marking system differ substantially from English, and an English-calibrated approach will not transfer without retraining on large Hindi corpora (Table 1).

Format-Based Methods

Format-based steganography encodes data by manipulating visual layout interword spacing, line height, character kerning, font weight, or paragraph alignment without altering textual content. These methods are text-content- agnostic but extremely fragile: any reformatting, copy-paste, or platform change destroys embedded data. For Devanagari documents in particular, format-based

methods fail even under normal editing, since Devanagari text rendering is highly sensitive to font variations and platform-specific shaping engine differences [3, 6].

Unicode-Based Methods

Unicode-based steganography exploits invisible characters such as Zero-Width Space (U+200B), Zero-Width Non-Joiner (U+200C), Zero-Width Joiner (U+200D), and Byte Order Mark (U+FEFF) to encode binary data between visible characters [12]. Alanazi et al. extended this approach to Arabic script, leveraging script-specific characters to increase capacity while maintaining visual naturalness [13].

For Devanagari, Unicode-based methods face two compounding problems. First, the Zero-Width Non-Joiner (ZWNJ, U+200C) and Zero-Width Joiner (ZWJ, U+200D) are not merely invisible characters in Devanagari they are semantically meaningful, controlling whether conjunct consonant clusters form or remain separated, directly affecting rendered glyphs. Inserting these characters as steganographic bits risks changing the visual appearance or even the linguistic meaning of the text. Second, Unicode normalisation (NFC and NFD) applied by standard text-processing software actively eliminates or reorders these zero- width characters, making this approach highly unreliable as a covert channel in Devanagari contexts.

OCR-Based Methods

OCR-based steganography encodes data in the rendered appearance of text and decodes it via optical character recognition. Higher embedding capacity is achievable because pixel-level character appearance can be manipulated more freely than Unicode representations. However, decoding accuracy depends directly on OCR system quality, image resolution, and font.

For Devanagari, OCR-based methods are particularly unsuitable. The best available Hindi OCR systems achieve 80–92% accuracy on clean print, compared to >99% for well-trained English OCR. Devanagari conjunct cluster complexity, context-sensitive matra positioning, and relative scarcity of large-scale Hindi OCR training data all contribute to this gap. Decoding errors in the OCR step translate directly into incorrect bit recovery, making Devanagari OCR steganography probabilistic rather than deterministic [14].

LLM-Based Generative Methods

LLM-based generative steganography encodes secret bits by controlling the token- sampling process of a large language model. Rather than modifying existing text, the system generates entirely new text in which each word choice encodes information. Since generated text follows the natural distribution of the language model, it is statistically difficult to distinguish from ordinary human-written text.

Lin et al. introduced zero-shot generative linguistic steganography combining LLM probability distributions with Huffman coding, achieving 1.9–4 bits per word without model fine-tuning [14]. Wang et al. proposed DAIRStega, which uses roulette-wheel interval mapping over the LLM probability distribution, reporting BLEU scores of 77 and topic divergence as low as 0.02 near-indistinguishable from human text [15]. Wu et al. demonstrated black-box LLM steganography operating via the public API of LLMs without access to model weights, widening deployment accessibility [16].

However, the steganalysis community has kept pace. Bai et al. showed that LoRA-fine-tuned language model detectors achieve 97% F1 accuracy in identifying LLM-generated steganographic text, revealing a fundamental arms-race dynamic [17]. Moreover, all state-of-the-art LLM-based methods are trained and evaluated exclusively on English; extension to Devanagari would require large- scale Hindi corpora, Devanagari-aware tokenisation, and appropriate evaluation metrics, none of which are currently standardised in the literature [18].

Glyph-Perturbation Methods

Glyph-perturbation steganography encodes data by applying minute, imperceptible changes to the geometric shapes of character glyphs within the font file sub-pixel shifts in Bézier control points,

stroke width alterations, or micro- curvature adjustments invisible to the human eye but machine-recognisable by a trained convolutional neural network (CNN).

FontCode is the landmark work in this category. It samples from a continuous font manifold a generative model of font shape variation and selects glyph variants that correspond to desired bit values according to a Chinese Remainder Theorem (CRT)-based encoding scheme. Each character has a codebook of 10–25 variants; five characters encode one byte. FontCode achieves 1.77 bits per character, survives print-scan conversion, and produces perceptual similarity scores above 0.85. These properties make it uniquely robust among text steganography systems.

The critical limitation is that the font manifold model was trained on Latin typefaces and its codebook is Latin-specific. Devanagari glyphs have fundamentally different geometric structure: strokes terminate with a horizontal headstroke (shirorekha), conjunct clusters form through ligature rules rather than juxtaposition, and the variant space of any given character is conditioned on the context of adjacent characters. Applying FontCode's Latin framework to Devanagari would require a new font manifold model trained on Devanagari typefaces, a new CNN classifier, and a new CRT-based codebook a substantial undertaking not yet addressed in the open literature.

Font-Metadata Methods

Font-metadata steganography encodes hidden information not in the rendered appearance of glyphs but in the internal metadata tables of the font file. TrueType and OpenType fonts store structured metadata in standardised tables including the name table (font family name, version, copyright, designer, and custom fields identified by NameID values), the head table (global font properties), and various feature tables used by shaping engines [19].

Custom NameID records in the range greater than 255 are user-defined. These records are preserved during normal font distribution, installation, and usage but are not examined or rendered by text layout engines. They represent a stable, deterministic covert channel with capacity bounded by the name table record storage limit (up to 65,535 bytes in standard TrueType/OpenType implementations). Unlike glyph-perturbation methods, font-metadata methods do not alter any glyph shape and are therefore completely independent of OCR, rendering quality, and font shaping rules including the complex context-sensitive shaping rules of Devanagari [20].

Lee et al. established the theoretical basis for font metadata as a covert channel and demonstrated proof-of-concept embedding in TrueType name tables [20]. Critically, the metadata channel is script-agnostic: since it is independent of the glyph tables, it works identically for any font including Devanagari fonts such as Noto Sans Devanagari or Mangal without requiring script-specific adaptation. This makes font-metadata methods the most immediately applicable approach for Devanagari steganography among all reviewed categories.

DEVANAGARI SCRIPT: STRUCTURAL PROPERTIES AND STEGANOGRAPHIC CHALLENGES

Devanagari is a left-to-right abugida used by over 600 million speakers, whose orthographic syllables, combining characters, and distinctive shirorekha provide unique opportunities as well as challenges for robust text steganography.

Script Architecture

Devanagari is an abugida (alphasyllabary) written left to right. It is the script for Hindi India's most widely spoken language as well as Marathi, Sanskrit, Nepali, Bodo, Maithili, Dogri, Konkani, Sindhi, and over 120 other languages, collectively used by approximately 600 million speakers and widely present in government, legal, academic, and digital communication across South Asia. Despite this

scale, Devanagari is dramatically underrepresented in steganography research compared to Latin and Arabic scripts.

The fundamental unit is the orthographic syllable, not the individual character. A syllable typically consists of a base consonant (or consonant cluster) possibly modified by a vowel diacritic (matra), a nasalisation mark (anusvara or chandrabindu), or a final consonant mark (visarga). The shirorekha a horizontal stroke running along the top of most characters, is a distinctive feature connecting letters within a word.

Properties That Break Existing Methods

Table 2 identifies the principal structural properties of Devanagari that cause standard text steganography methods to fail or perform poorly.

These properties interact. A method resilient to one challenge alone may fail when multiple challenges are present simultaneously. For example, a format-based method using inter-character spacing may work within the same platform but fail when the document is opened on a different operating system with a different Devanagari shaping engine (Table 2).

The Research Gap

A systematic search of text steganography literature from 2020 to 2026 reveals that no published work presents a high-capacity, robust, and deterministic text steganography system specifically designed and evaluated on Devanagari script. Banerjee and Choudhury surveyed steganography for Indic scripts and confirmed this gap, noting that while matra-shifting and Katapayadi numeral encoding had been explored in earlier work, no modern high-capacity system had been developed. These earlier approaches achieved only 1–2 bits per character and were highly dependent on font rendering.

The practical implications are significant. Indian government documents, legal archives, academic publications, and official communications are extensively produced in Devanagari. The absence of a robust Devanagari steganography system means that covert communication and document watermarking for this enormous body of text is technically unsupported.

Table 1. Standard evaluation metrics for text steganography systems.

Metric	Definition	Typical measurement
Embedding capacity	Hidden data per unit of cover text	Bits/character (bpc) or bits/word (bpw)
Imperceptibility	Visual/linguistic indistinguishability	Human rating (>0.85), PSNR
Robustness	Survives text transformations	Recovery rate (%) after reformatting/OCR
Determinism	Consistent decoding across runs	Exact match rate over *N* runs
Steganalysis resistance	Resists ML-based detection	AUC, F1-score of detector, KL divergence
Computational efficiency	Encoding/decoding resource cost	Wall-clock time (MS), memory (MB)
Script awareness	Correct target-script handling	Qualitative assessment, glyph/cluster correctness

Table 2. Structural challenges of Devanagari script for text steganography.

Challenge	Description	Impact
Conjunct consonants	Multiple consonants fused into one glyph	Glyph segmentation unreliable for OCR methods
Matra (diacritics)	Vowel markers above/below base consonant	Positional shifts alter meaning
Halant character	Virama → suppresses vowel, forms clusters	Cluster boundary detection difficult
Non-linear rendering	Non-sequential visual composition rendering methods fail	Rendering-dependent
NFC/NFD	Collapses variant code points	Unicode-variant channels stripped

normalisation		
Font substitution	Different fonts render same code points differently	Glyph codebooks become invalid
OCR sensitivity	Accuracy ∼80–90% vs >99% Latin	Higher bit-error rates

FONT-BASED STEGANOGRAPHY: A CLOSER EXAMINATION

Font-based steganography whether glyph-perturbation or font-metadata operates on the font file rather than on rendered text. This architectural choice has a fundamental implication for Devanagari: the steganographic channel is decoupled from the rendering pipeline. Since hidden data is stored in the font file rather than in the rendered output, it is not subject to the shaping rules, normalisation processes, or OCR dependencies that affect rendering-dependent methods.

Why Font-Based Methods Suit Devanagari

A Devanagari font file contains glyph outlines alongside OpenType GSUB (glyph substitution) and GPOS (glyph positioning) tables that implement conjunct formation, matra attachment, and other shaping operations. A font- metadata channel embedded in the name table of such a font is completely separate from all of these: it resides in the font binary but has no influence on and receives no influence from the rendering process. This makes it inherently robust to Devanagari's complex shaping rules.

Glyph-Perturbation: Status and Requirements for Devanagari

FontCode remains the state of the art in glyph-perturbation steganography. Its pipeline consists of: font manifold sampling (generating a continuous space of glyph variants); perceptual filtering (retaining only variants within the human perceptual threshold); CNN filtering (retaining only variants reliably machine- distinguishable); codebook construction (assigning numeric codes to retained variants); and CRT-based encoding/decoding with error correction [21].

Extending this pipeline to Devanagari would require:

1. A generative model of the Devanagari font manifold trained on a large corpus of Devanagari typefaces;
2. A CNN classifier trained on Devanagari glyph variants with specific handling for context-dependent glyph forms;
3. A new CRT-based codebook calibrated to the number of distinguishable Devanagari glyph variants per character; and
4. Integration with OpenType GSUB tables to ensure that substituted glyph variants do not trigger unintended conjunct formations or shaping errors. Each of these is a non-trivial research contribution.

GAN-based glyph synthesis which has shown promise for Latin and CJK typefaces represents a promising approach to addressing component [22].

Font-Metadata: Advantages and Limitations

Font-metadata steganography embeds secret messages in the internal metadata tables of font files rather than in rendered glyph shapes. The encoding mechanism is straightforward: a secret message is converted to a binary bitstream, prefixed with a length header, and stored as a string in a custom NameID record (range>255) of the font's name table.

The principal advantages are:

1. The channel is completely deterministic given the same stego font, the same message is always recovered.
2. It is OCR-free and rendering-independent.
3. It is script-agnostic, working identically for any font including Devanagari fonts.
4. Capacity is high, bounded only by name table storage (up to 65,535 bytes per record, supporting approximately 21,845 three-byte Devanagari characters in UTF-8 encoding).

The principal limitations are:

1. The covert channel is destroyed by active font sanitisation (FontForge re-save, metadata stripping, font format conversion, or PDF export with font subsetting).
2. An adversary with binary-level access to the font file can inspect the name table and detect non-standard NameID records.
3. The channel requires that the same font file be shared between sender and receiver. These limitations are consistent with the passive-adversary threat model that governs most practical covert communication scenarios. Pre-embedding encryption with AES-256 addresses limitation (2) partially, protecting message content even if the covert channel is detected. Redundant embedding across multiple name table records combined with Reed-Solomon error-correcting codes can improve resilience against partial font sanitisation.

COMPARATIVE ANALYSIS

Table 3 presents a chronological summary of representative works reviewed in this paper, and Table 4 provides a category-level comparison across the standard evaluation metrics established in Section 2.

Several observations emerge from this analysis. First, there is a clear trajectory from low-capacity heuristic methods towards high-capacity neural and font- internal approaches. LLM-based generative methods currently achieve the highest embedding rates (up to 4 bits per word) with strong statistical imperceptibility but require shared language models and remain exclusively English-language. (Table 3) Second, font-based methods are the only category that is in principle script- agnostic, making them the natural choice for Devanagari. Third, no method in the review period explicitly addresses Devanagari, confirming the research gap identified in Section 4. Fourth, steganalysis capability has grown substantially, with LoRA-tuned LLM detectors reaching 97% F1 accuracy, placing new pressure on the imperceptibility requirements of all embedding approaches (Table 4).

Research Gaps and Future Directions

Table 5 summarises the principal research gaps and corresponding future directions identified by this review.

Devanagari Font Manifold for Glyph Perturbation

The most technically demanding open problem is constructing a Devanagari font manifold analogous to that used in FontCode. Such a manifold would parameterise the continuous space of perceptually acceptable Devanagari glyph shapes. Training requires a large corpus of Devanagari typefaces the Google Fonts library and CDAC India font collections are potential sources and a generative model architecture sensitive to Devanagari stroke topology. GAN-based glyph synthesis, which has proven effective for Latin and CJK typefaces, is a natural methodological candidate. (Table 5).

Multi-Script Unified Framework

A generalised font-based steganography framework accommodating Latin, Devanagari, Tamil, Telugu, Bengali, and Arabic within a unified codebook and encoding architecture would be a substantial contribution. The font-metadata approach is the most viable foundation for such a framework, since it is intrinsically script-independent. The glyph-perturbation approach would require per-script font manifolds and CNN classifiers but could share the CRT-based encoding layer.

Standardised Devanagari Evaluation Benchmarks

The absence of standardised evaluation datasets for Devanagari steganography is itself a significant gap. A benchmark corpus analogous to English steganalysis datasets built from the Hindi Wikipedia, government document archives, or Devanagari books would enable reproducible cross-system comparison. Such benchmarks should include adversarial test cases for Unicode normalisation, font substitution, and platform-specific rendering differences.

Steganalysis for Font-Based Channels

While linguistic steganalysis for LLM- generated text has advanced rapidly, steganalysis for font-based covert channels remains underdeveloped. Font comparison tools can detect binary differences against a known clean baseline, but no systematic automated steganalyser for font-metadata or glyph-perturbation channels has been published. Developing such tools and using them to calibrate the security of embedding approaches is an important open direction.

Table 3. Chronological summary of representative text steganography works (2017– 2026).

Year	Work	Method	Cap.	Key Finding / Gap
2017	Xiao et al.	Font-Glyph Code + CRT + CNN	1.77 bpc	Print-resilient; Latin only
2019	Ziegler et al.	GPT-2 arithmetic	1–5 bpw	High naturalness; English only
2020	Alanazi et al.	Unicode Arabic	∼2 bpc	Script-aware; Arabic-specific
2021	Yang et al.	Variational AE (VAE-Stega)	∼3 bpw	Improved security; Latin only
2022	Xiang et al. (survey)	LLM review	N/A	No Indic coverage identified
2022	Banerjee & Choudhury	Indic survey	N/A	Confirmed Devanagari gap
2023	Ding et al.	Distribution-pres. LLM	∼4 bpw	SOTA security; English only
2024	Lin et al.	Zero-shot LLM	1.9–4 bpw	Scalable; English-centric
2024	Wu et al.	Black-box LLM	∼3 bpw	API-compatible; English only
2024	Bai et al.	LoRA detector	N/A	97% F1; arms-race confirmed
2025	Huang et al.	Markov DPS	∼3 bpw	Provably secure; English only
2025	Wang et al.	Roulette-wheel DAIRStega	∼4 bpw	BLEU=77; scalability TBD

Table 4. Category-level comparison of text steganography approaches.

Category	Capacity	Robust.	Script	Key Limitation
Linguistic	Low (0.5–2 bpw)	Low	Lang.-dep.	Semantic drift; English-centric
Format	<1 bpc	Very low	No	Destroyed by reformat
Unicode	1–2 bpc	Medium	Partial	Stripped by NFC/NFD; ZWJ semantic in Devanagari
OCR-based	2–4 bpc	Low	Sensitive	Probabilistic; 80–90% on Devanagari
Glyph-perturb.	∼1.77 bpc	High	Latin only	CNN/codebook Latin-specific
Font-metadata	High (file-lim.)	High	Yes	Sanitisation destroys channel
LLM-generative	1.9–4 bpw	Medium	Limited	97% F1 detection; English only

Table 5. Research and future directions in font-based Devanagari steganography.

Direction	Description
Multi-script font metadata	Extend font-internal channels to Tamil, Telugu, Bengali, Arabic
GAN-based glyph variants	Auto-generate imperceptible Devanagari glyph pairs
Hybrid semantic-visual	Combine LLM paraphrasing with glyph-level perturbation
Error-correcting codes	Apply Reed-Solomon ECC to embedded bitstream
Adversarial steganalysis	Train against CNN/transformer font forensic detectors
Standardised Indic benchmarks	Hindi Wikipedia-based evaluation corpus
Encryption by default	AES-256 pre-embedding as standard pre-processing
Editor integration	Embed steganographic logic into text editors

Practical Deployment

Real-world deployment faces additional challenges: secure distribution of modified font files, compatibility with web font formats (WOFF and WOFF2), font subsetting behaviour during PDF

export, and integration into document creation workflows. Each represents an applied research direction complementing the foundational technical work.

CONCLUSION

This paper has reviewed text steganography techniques from 2020 to 2026, with a specific focus on font-based approaches and their applicability to Devanagari script. The review classified methods into seven categories linguistic, format-based, Unicode-based, OCR-based, glyph-perturbation, font-metadata, and LLM-generative and analysed them across a unified framework of evaluation metrics comprising capacity, imperceptibility, robustness, determinism, steganalysis resistance, computational efficiency, and script awareness.

The central finding is that despite the enormous scale of Devanagari's digital presence hundreds of millions of speakers and pervasive use in government, law, and academia across South Asia no high-capacity, robust, and deterministic text steganography system has been published for this script. The structural properties of Devanagari (conjunct consonants, context-sensitive glyph composition, matra placement, and Unicode normalisation behaviour) cause rendering-dependent and OCR-based methods to fail, while LLM-based generative approaches remain exclusively English-language.

Font-based methods specifically font-metadata encoding emerge from this review as the most promising technical direction for Devanagari steganography. Embedding data in the internal metadata tables of TrueType/OpenType font files achieves deterministic, OCR-free, and script-agnostic covert communication without altering glyph appearance. Glyph-perturbation methods, while offering higher print-scan robustness, require the construction of a Devanagari- specific font manifold, CNN classifier, and codebook a substantial but tractable undertaking that GAN-based glyph synthesis may help to address.

The most urgent priorities for the research community are constructing a Devanagari font manifold for glyph-level steganography; developing standardised Devanagari evaluation benchmarks; extending font-metadata methods with error- correcting codes and default encryption; and conducting systematic steganalysis of font-based covert channels. These directions would close the substantial gap that currently exists between the state of the art in Latin-script steganography and the needs of the world's second most widely written script.

REFERENCES

1. Rani N, Chaudhary J. Text steganography techniques: A review. International Journal of Engineering Trends and Technology (IJETT). 2013 Jul;4(7):3013-3015.
2. Liu A, Pan L, Lu Y, Li J, Hu X, Zhang X, Wen L, King I, Xiong H, Yu P. A survey of text watermarking in the era of large language models. ACM Computing Surveys. 2024 Nov 7;57(2):1-36.
3. Majeed MA, Sulaiman R, Shukur Z, Hasan MK. A review on text steganography techniques. Mathematics. 2021 Nov 8;9(21):2829.
4. Al-Ghaili AM, Kasim H, Hassan Z, Al-Hada NM, Othman M, Kasmani RM, Shayea I. A review: image processing techniques' roles towards energy-efficient and secure iot. Applied sciences. 2023 Feb 6;13(4):2098.
5. Ahvanooey MT, Zhu MX, Li Q, Mazurczyk W, Choo KK, Gupta BB, Conti M. Modern authentication schemes in smartphones and IoT devices: An empirical survey. IEEE Internet of Things Journal. 2021 Dec 24;9(10):7639-63.
6. Vishwanath NV, Manjunathachari K, Prasad KS. Multi-lingual character segmentation and recognition based on adaptive projection profiles and composite feature vectors. Multimedia Tools and Applications. 2023 Jul;82(16):24247-68.
7. Xiao C, Zhang C, Zheng C. Fontcode: Embedding information in text documents using glyph perturbation. ACM Transactions on Graphics (TOG). 2018 Feb 28;37(2):1-6.

8. Simmons GJ. The prisoners' problem and the subliminal channel. In Advances in Cryptology: Proceedings of Crypto 83 1984 Aug 22 (pp. 51-67). Boston, MA: Springer US.
9. Li S, Wang J, Liu P. Detection of generative linguistic steganography based on explicit and latent text word relation mining using deep learning. IEEE Transactions on Dependable and Secure Computing. 2022 Mar 10;20(2):1476-87.
10. Yang ZL, Zhang SY, Hu YT, Hu ZW, Huang YF. VAE-Stega: Linguistic Steganography based on variational autoencoder. IEEE Transactions on Information Forensics and Security. 2020 Sep 10; 16:880-95.
11. Zhou X, Peng W, Yang B, Wen J, Xue Y, Zhong P. Linguistic steganography based on adaptive probability distribution. IEEE Transactions on Dependable and Secure Computing. 2021 May 13;19(5):2982-97.
12. Alanazi N, Khan E, Gutub A. Involving spaces of unicode standard within irreversible Arabic text steganography for practical implementations. Arabian Journal for Science and Engineering. 2021 Sep;46(9):8869-85.
13. Lin K, Luo Y, Zhang Z, Ping L. Zero-shot generative linguistic steganography. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers) 2024 Jun (pp. 5168-5182).
14. Yang B, Peng W, Xue Y, Zhong P. A Generation-based Text Steganography by Maintaining Consistency of Probability Distribution. KSII Trans. Internet Inf. Syst. 2021 Nov 1;15(11):4184-202.
15. Cao Y, Zhou Z, Chakraborty C, Wang M, Wu QJ, Sun X, Yu K. Generative steganography based on long readable text generation. IEEE transactions on computational social systems. 2022 May 19;11(4):4584-94.
16. Zhang Z, Wen J, Gao L, Peng W, Xue Y. Linguistic steganalysis based on few-shot adversarial training. IEEE Transactions on Dependable and Secure Computing. 2025 Jan 22;22(4):3514-28.
17. KL SN, R BK. Text steganography: enhanced character-level embedding algorithm using font attribute with increased resilience to statistical attacks. Multimedia Tools and Applications. 2025 Mar;84(11):9367-92.
18. Meng J, Lu Z. Advances in Semantic-Preserving Text Watermarking. Sensors. 2026 Feb 28;26(5):1528.
19. Ji Y, Cao J, Li Q, Liu Y, Wei T, Xu K, Wu J. Constructing SDN covert timing channels between hosts with unprivileged attackers. IEEE Transactions on Networking. 2024 Nov 20;33(2):612-23.
20. Petitcolas FA, Anderson RJ, Kuhn MG. Information hiding-a survey. Proceedings of the IEEE. 1999 Jul 31;87(7):1062-78.
21. Yang ZL, Guo XQ, Chen ZM, Huang YF, Zhang YJ. RNN-stega: Linguistic steganography based on recurrent neural networks. IEEE Transactions on Information Forensics and Security. 2018 Sep 24;14(5):1280-95.
22. Tran TM, Vu TN, Vo ND, Nguyen TV, Nguyen K. Anomaly analysis in images and videos: A comprehensive review. ACM Computing Surveys. 2022 Dec 15;55(7):1-37.

