DiffNeCTIS: Diffusion-based Nested Compound Type Identification for Sanskrit

Anonymous ACL submission

Abstract

Sanskrit multi-component compounding presents a severe structured prediction challenge: the space of valid parse structures grows exponentially with compound length, implicit semantic relations are highly context-sensitive, and accurate type assignment demands simultaneous global structural coherence and local morphological precision across a fine-grained label typology. Existing systems address Nested Compound Type Identification for Sanskrit (NeCTIS) discriminatively, predicting dependency edges and semantic types in a single greedy pass. Reformulating NeCTIS as iterative generative refinement via diffusion over the full label sequence resolves this structural inconsistency but exposes a complementary bottleneck - coarse global attention that suppresses the localized morphological signals necessary for fine-grained type discrimination. We address this through a local refinement mechanism within a restricted window.

We further provide the first systematic evaluation of large language models on NeCTIS, finding consistent and substantial failures under zero-shot and few-shot conditions - attributable to the absence of structural inductive biases, Sanskrit under-representation in pretraining, and morphological sensitivity requirements that broad-coverage models are fundamentally ill-equipped to satisfy.

1 Introduction

Sanskrit, a morphologically rich and highly fusional language, is renowned for its extensive use of compounding (*Samāsa*) (Coulson, 1976; Wright, 1968). Unlike structurally rigid languages where compounds are typically binary, Sanskrit texts routinely employ multi-word, deeply nested compounds to express complex semantic propositions. According to the Digital Corpus of Sanskrit (Hellwig, 2010–2021), more than 41% of compounds contain three or more components (Krishna et al., 2016), and in classical poetry their use is near-ubiquitous (Kumar and Kulkarni, 2012). Decoding such compounds demands resolving a two-fold puzzle: identifying the precise boundaries of the nested constituent spans, and decoding the implicit semantic relations between them. This challenge has been formalized as the Nested Compound Type Identification (NeCTIS) task, with the DepNeCTI benchmark (Sandhan et al., 2023b) providing evaluation datasets and strong discriminative baseline.. As illustrated in Figure 1, the compound *sumitrā-Ānanda-Vardhanaḥ* ("Sumitrā-delight-enhancer"), an example of the NeCTIS task, highlighting the nested spans and their associated semantic relations using distinct colors.

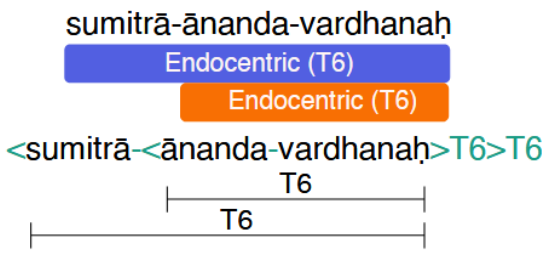

Figure 1: llustration of the NeCTIS task for the multi-component compound Sumitrā-Ānanda Vardhanaḥ (Translation: Sumitra-delight-enhancer), highlighting nested spans and their associated semantic relations using distinct colors.

Existing NeCTIS systems rely on discriminative Biaffine parsers that predict span boundaries and semantic types in a single greedy pass (Dozat and Manning, 2017; Kitaev et al., 2019; Sandhan et al., 2023b). Although effective, these models are mainly based on locally computed edge scores, which may limit their ability to capture global structural dependencies in deeply nested compounds (McDonald et al., 2005; Zhang et al., 2020). Furthermore, Large language models provide a natural

alternative, but their autoregressive nature does not explicitly enforce hierarchical structure. Moreover, Sanskrit is underrepresented in most pretraining corpora, which may limit their effectiveness on fine-grained compound type identification. Our experiments confirm that current LLMs perform poorly on NeCTIS under few-shot settings and instruct fine-tuned settings.

We reformulate NeCTIS as a generative structured prediction problem, instantiated via **DiffusionSL** (Huang et al., 2023), a discrete denoising diffusion framework operating over continuous bit-space label representations. By iteratively refining the *full* label sequence from noise, the model conditions each denoising step on the global label state, naturally suppressing globally inconsistent structures. Global self-attention distributes the representational capacity across all tokens, which can dilute the influence of localized morphological cues when compounds contain many components. We refer to this phenomenon as the *Granularity Gap*, where the global context dominates over the local morphological signals required for fine-grained type discrimination (Bohnet et al., 2013; Hellwig, 2016; Lai et al., 2025). We address this with a local refinement module, a restricted-window attention mechanism that restores sensitivity to $\mathcal{O}(1/w)$ for window $w \ll n$- recovering fine-grained discriminative capacity within the globally consistent generative framework.

Experiments on DiffNeCTIS demonstrate improvements in Unlabeled Span Score (USS) and Labeled Span Score (LSS) and Exact Match (EM) over strong discriminative baselines. Critically, we observe that local refinement induces a systematic tradeoff: EM gains are accompanied by compression in USS and LSS, which we attribute to probability mass concentration toward high-confidence structural patterns at the expense of span-level recall on rarer compound types, a finding with broader implications for diffusion-based structured prediction in morphologically rich languages. Our contributions are as follows:

- We reformulate NeCTIS as a diffusion-based structured prediction problem.
- We provide the first systematic LLM evaluation on NeCTIS, diagnosing failure along structural, linguistic, and discriminative axes.
- We identify the *Granularity Gap* in diffusion-based sequence labelling and propose a local refinement module that recovers fine-grained morphological discrimination within a globally consistent generative framework.

2 Background

2.1 Sanskrit Compounding and the NeCTIS Task

Sanskrit compounding (*Samāsa*) combines two or more nominal stems into a single word, requiring euphonic transformations (*sandhi*) at constituent borders. This sandhi process blurs surface constituent boundaries, making segmentation necessary for downstream analysis. The meaning of a compound comes from the semantic compatibility (sāmarthya) between its constituents, as defined in the Pāṇinian grammatical tradition (Hellwig and Petersen, 2012). A typology of compound classes, including *Tatpuruṣa* (endocentric/determinative), *Bahuvrīhi* (exocentric), *Dvandva* (copulative), *Karmadhāraya* (appositional), and *Avyayībhāva* (indeclinable), yields up to 56 fine-grained subtypes (Ramkrishnamacharyulu et al., 2012)

Sanskrit Compound Type Identification (SaCTI). Sanskrit Compound Type Identification has attracted significant attention over the past decade. Decoding the meaning of a Sanskrit compound requires determining its constituents (Huet, 2009; Hellwig and Nehrdich, 2018), understanding how these constituents are grouped and identifying the semantic relationship between them (Kumar and Kulkarni, 2012). Previous studies proposed rule-based approaches (Kulkarni and Dangarikar, 2013), a data-driven approach (Sandhan et al., 2023a), and a hybrid approach (Krishna et al., 2016) for SaCTI. Sandhan et al. (2022) proposed a context-sensitive architecture for binary compounds.

Nested Compound Type Identification for Sanskrit (NeCTIS). Sandhan et al. (2023b) introduced the Nested Compound Type Identification task to extend compound analysis to the multi-component setting. Multi-component compounds are formed through successive binary combinations of constituents, producing a nested hierarchical structure whose parse space grows combinatorially. A compound of N components admits C_{N-1} valid parse structures, where C_n is the n-th Catalan number (Huet, 2009):

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\,n!} \qquad (1)$$

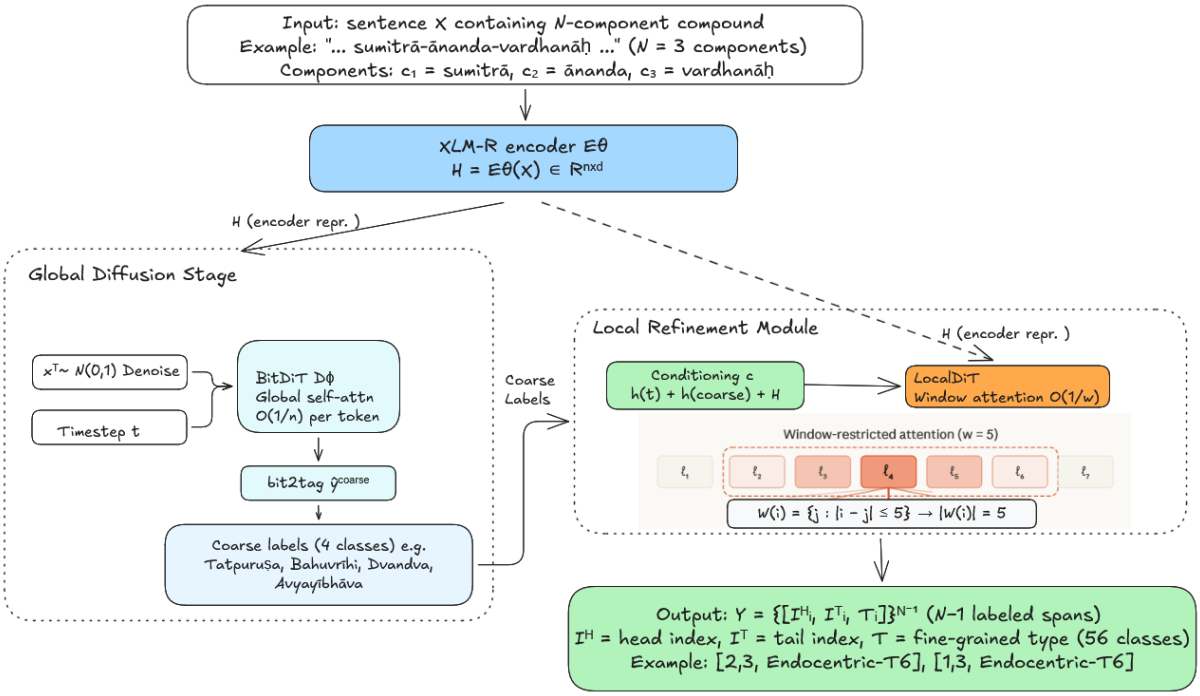

Figure 2: DiffNeCTIS architecture with Local Refinement. The global stage denoises coarse label sequences via BitDiT with full self-attention; the local refinement stage applies window-restricted attention ($w = 5$) to recover fine-grained type distinctions suppressed by $\mathcal{O}(1/n)$ signal dilution.

Formally, given a sentence $X = \{x_1, x_2, \ldots, x_M\}$ containing an N-component compound x_p, the NeCTIS system must produce:

$$Y_p = \{\, [I_i^H,\, I_i^T,\, T_i] \,\}_{i=1}^{N-1}, \quad Y_p \in P_N \tag{2}$$

where I_i^H and I_i^T denote the head and tail indices of the i-th span, T_i is the compound type label, and P_N is the set of all valid full parenthesizations with $|P_N| = C_{N-1}$. The task assumes prior knowledge of compound boundaries and segmented components.

2.2 Diffusion Models for Structured Prediction

Denoising Diffusion Probabilistic Models (DDPM; Ho et al., 2020) define a forward process that injects Gaussian noise into clean data $\mathbf{x}_0$ over T timesteps, yielding $\mathbf{x}_t \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0,\, (1-\bar{\alpha}_t)\mathbf{I})$, and a learned reverse process $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ that recovers the data via iterative denoising. All positions are denoised in parallel, avoiding the exposure bias of autoregressive decoders (Arora et al., 2022).

Applying diffusion to sequence labeling requires bridging discrete tags with continuous noise. Diffusion based sequence labeling architectures (Huang et al., 2023; Shen et al., 2023; Yin et al., 2025; Han et al., 2023) address this via a Bit-Tag Converter (**BTConverter**) that encodes each tag as its $\lceil \log_2 |\mathcal{S}| \rceil$-bit binary index scaled to a continuous range, and a Bit Diffusion Transformer (**BitDiT**) that models the conditional reverse process with DDIM sampling (Song et al., 2022).

3 Proposed Method

3.1 Problem Formulation

We formalize Nested Compound Type Identification (NeCTIS) as a conditional structured prediction problem over hierarchically nested spans. the NeCTIS system must produce a list of $N-1$ labeled spans:

$$Y_p = \{\, [I_i^H,\, I_i^T,\, T_i] \,\}_{i=1}^{N-1} \tag{3}$$

The count $N-1$ follows from the binary-branching structure of Sanskrit compounding: combining components N through successive binary merges yields exactly $N-1$ internal nodes in the resulting parse tree, each requiring a span and a type label, making NeCTIS a joint structure-label prediction problem. I_i^H and I_i^T denote the head and tail component indices of the i-th span, and $T_i \in \mathcal{T}$ is the compound type label assigned to that span, drawn from a fine-grained taxonomy of $|\mathcal{T}| = 56$ classes.

A valid output Y_p must correspond to a full binary parenthesization of the N components. Let

$\mathcal{P}_N$ denote the set of all valid parenthesizations; its cardinality satisfies $|\mathcal{P}_N| = C_{N-1}$, where $C_{N-1} = \frac{1}{N}\binom{2(N-1)}{N-1}$ is the $(N-1)$-th Catalan number.

We define the training objective as conditional structured prediction. Given a training set $\mathcal{D} = \{(X^{(i)}, Y^{(i)})\}_{i=1}^{|\mathcal{D}|}$, we seek parameters θ that maximize:

$$\theta^* = \arg\max_{\theta} \sum_{j=1}^{|\mathcal{D}|} \log p_\theta\left(Y^{(i)} \mid X^{(i)}\right) \quad (4)$$

Because each compound's output must lie in $\mathcal{P}_N \times \mathcal{T}^{N-1}$, the model must capture both the structural constraints governing valid parenthesizations and the fine-grained semantic distinctions that determine type labels.

3.2 Diffusion-based Sequence Labeling for NeCTIS

We cast NeCTIS as a diffusion-based sequence labeling task, adapting the DiffusionSL framework (Huang et al., 2023) to generate compound type label sequences via iterative denoising. Each span label $T_i \in \mathcal{T}$ is mapped to a continuous bit representation via `tag2bit`*: given $|\mathcal{T}| = 56$ classes, each label is encoded as a $\lceil \log_2 56 \rceil = 6$-bit binary vector scaled to $[-b, b]^6$, yielding clean data $\mathbf{x}_0 \in \mathbb{R}^{(N-1)\times m}$. Algorithms 1 and 2 describe the training and inference procedures, respectively.

The decoder $\mathcal{D}_\phi$ employs a Transformer-based architecture (BitDiT) §A.2 with adaptive layer normalization to incorporate $\mathbf{w}$ and the timestep embedding. Because $\mathcal{D}_\phi$ applies self-attention across all $N-1$ label positions at every timestep, it maintains a global receptive field over the full label structure, enabling the model to capture long-range dependencies among span labels and ensure structurally consistent parenthesizations. We use the following notation throughout: $\mathbf{H} \in \mathbb{R}^{M\times d}$ denotes the encoder output $\mathcal{E}_\theta(X)$; $t \in \{1, \ldots, T\}$ is the diffusion timestep; $\mathbf{x}_t \in \mathbb{R}^{(N-1)\times m}$ is the noised label representation at step t; $\hat{y}^{\text{coarse}}$ is the coarse-label prediction from the global stage; $\mathbf{c}$ is the conditioning signal aggregating $\mathbf{H}$, t, and $\hat{y}^{\text{coarse}}$; and LocalDiT is the window-attention denoiser of the local stage. $\mathcal{L}_{\text{diff}}^{\text{global}}$ denotes the standard DDPM noise-prediction loss on the global stage.

The Granularity Gap. BitDiT's global self-attention enables structurally consistent parenthesizations, but bounds each token's contribution at $\mathcal{O}(1/n)$ (Hahn, 2020) at every denoising step. Coarse distinctions like *tatpuruṣa* vs. *bahuvrīhi* survive this dilution since they rely on aggregate semantics, but fine-grained ones do not: separating *ṣaṣṭhī-tatpuruṣa* (genitive) from *saptamī-tatpuruṣa* (locative) depends on the head-modifier pair, and the case-marking suffixes that carry this signal are adjacent surface cues (Tsarfaty et al., 2013; Hellwig, 2016; Sandhan et al., 2023b) that get washed out as n grows toward 16. We therefore split denoising into two stages: a global stage for coarse structure ($\text{GlobalDiT}(\mathbf{x}_t,\ t,\ \mathbf{H})$) and a local stage with window-restricted attention to concentrate signal at head-modifier boundaries ($\text{LocalDiT}(\mathbf{x}_t,\ t,\ \mathbf{H},\ \hat{y}^{\text{coarse}})$; subsection 3.3).

3.3 From Coarse to Fine: Window-Restricted Denoising

To address the granularity gap, we augment the diffusion denoising process with a local refinement stage that restores $\mathcal{O}(1/w)$ signal strength for pairwise component interactions, directly targeting the fine-grained label distinctions that global self-attention dilutes. The algorithms are shown in 1 and 2.

Local Refinement via Window Attention. We replace full self-attention in the refinement stage with window-restricted attention. For a window of size w, the local neighborhood of position i is defined as $\mathcal{W}(i) = \{j : |i-j| \le \lfloor w/2 \rfloor\}$, and the local attention output is:

$$\text{LocalAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})_i = \sum_{j\in\mathcal{W}(i)} \alpha_{ij}\, \mathbf{v}_j \quad (5)$$

where the attention weights are computed with a windowed softmax:

$$\alpha_{ij} = \begin{cases} \dfrac{\exp\left(\mathbf{q}_i^\top \mathbf{k}_j/\sqrt{d}\right)}{\sum_{l\in\mathcal{W}(i)} \exp\left(\mathbf{q}_i^\top \mathbf{k}_l/\sqrt{d}\right)} & j \in \mathcal{W}(i) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Since $|\mathcal{W}(i)| = w$ is independent of sequence length n, the expected per-token influence is $\mathcal{O}(1/w)$ rather than $\mathcal{O}(1/n)$, and the sensitivity of the local representation $\mathbf{z}_j$ to token x_i satisfies:

$$\left\| \frac{\partial \mathbf{z}_j}{\partial x_i} \right\| = \begin{cases} \mathcal{O}(1) & |i-j| \le \lfloor w/2 \rfloor \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

This $\mathcal{O}(1)$ bound, independent of n, directly addresses the signal dilution identified in (Hahn,

2020). With $w = 5$, each position attends to its two nearest neighbors on either side, capturing the local head-modifier relationships critical for fine-grained type distinction.

Conditioned Local Denoising. The local refinement stage operates as a secondary denoising network conditioned on the encoder representations and the coarse predictions from the global diffusion stage. Given coarse prediction $\hat{y}^{\text{coarse}}$, the local denoiser predicts the fine-grained clean labels:

$$\hat{\mathbf{x}}_0^{\text{fine}} = \text{LocalDiT}(\mathbf{x}_t,\ t,\ \mathbf{H},\ \hat{y}^{\text{coarse}}) \quad (8)$$

The conditioning signal aggregates temporal, structural, and contextual information:

$$\mathbf{c} = \mathbf{h}_{\text{time}}(t) + \mathbf{h}_{\text{coarse}}(\hat{y}^{\text{coarse}}) + \mathbf{H} \quad (9)$$

where $\mathbf{h}_{\text{time}}(t)$ is a sinusoidal timestep embedding and $\mathbf{h}_{\text{coarse}}(\hat{y}^{\text{coarse}})$ is a learned embedding of the predicted coarse category. Conditioning on $\hat{y}^{\text{coarse}}$ constrains the effective label space to fine-grained types consistent with the predicted coarse category, reducing search complexity from a flat 56-class prediction to approximately $\max_c |F_c|$ classes per coarse group c.

The overall training objective combines the global and local stage losses:

$$\mathcal{L} = \mathcal{L}_{\text{diff}}^{\text{global}} + \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[\left\|\hat{\mathbf{x}}_0^{\text{fine}} - \mathbf{x}_0\right\|^2\right] \quad (10)$$

Design Trade-offs. Window-restricted attention improves fine-grained label precision but limits the receptive field available for structural decisions. Span boundary identification (measured by USS) requires awareness of global compound structure (Kitaev et al., 2019; Beltagy et al., 2020), particularly for long compounds where distant components influence parenthesization. Restricting attention to a fixed window may therefore reduce structural accuracy in exchange for label precision. Additionally, conditioning on a fixed $\hat{y}^{\text{coarse}}$ might introduce a cascading dependency: errors in the global stage propagate into the local stage, potentially degrading USS and LSS when coarse predictions are unreliable (Finkel et al., 2006). We analyze these trade-offs empirically in §5.4.

4 Experimental Settings

4.1 Datasets

We evaluate on the NeCTIS benchmark (Sandhan et al., 2023b), comprising an in-domain dataset (33,055 compounds; 22,547/4,107/6,401 train/dev/test) and an out-of-domain test set (NeCTIS OOD; 1,189 compounds drawn from different source texts). Compounds are pre-segmented and range from 2 to 16 components, with annotations at two granularities: coarse (4 types) and fine-grained (56 sub-types). The distribution is heavily skewed toward binary compounds and *Tatpuruṣa* sub-types. The combination of exponential parse space (Catalan number growth), long-tail label distribution, and context-dependent structural ambiguity makes NeCTIS a challenging low-resource benchmark. All models are trained on the NeCTIS training split Table 1 and results Table 2 are averaged over 3 runs.

Dataset	#Comp.	Train	Dev	Test	#Types
NeCTIS	33055	22547	4107	6401	4 (56)
NeCTIS OOD	1803	–	–	1803	4 (56)

Table 1: Data statistics for NeCTIS and NeCTIS-OOD

4.2 Evaluation Metrics

We adopt the three metrics from (Sandhan et al., 2023b):

Unlabeled Span Score (USS). Micro-averaged F1 over predicted spans, ignoring type labels. USS measures structural prediction accuracy: whether the model identifies the correct nested parenthesization of components.

Labeled Span Score (LSS). Micro-averaged F1 over span–label tuples (I_i^H, I_i^T, T_i). A predicted span is counted as correct only if both the span boundaries and the compound type label match the gold annotation. LSS captures joint structural and semantic accuracy.

Exact Match (EM). The percentage of compounds for which *all* $N-1$ predicted span-label tuples exactly match the gold parse. EM is the strictest metric, reflecting full parse correctness and penalizing any single span or label error.

4.3 Baseline Methods

We compare against systems spanning two modeling paradigms namely *Large Language Modelling* and *Discriminative Paradigms*, chosen to isolate the contributions of encoding architecture, training objective, and decoding strategy.

Large Language Models. We evaluate whether explicit linguistic rules help LLMs perform Sanskrit compound analysis on NeCTIS (Table 3), under both coarse- and fine-grained settings with

System	Coarse						Fine-grained					
	w/o context			w/ context			w/o context			w/ context		
	USS	LSS	EM	USS	LSS	EM	USS	LSS	EM	USS	LSS	EM
NeCTIS												
LSTM – CE	95.46	89.06	76.82	97.42	89.24	77.00	95.38	79.72	56.83	97.25	79.56	57.98
LSTM – CSSL	95.24	87.71	74.48	97.52	89.95	78.23	95.67	80.55	59.18	98.18	80.79	59.25
XLMR – CE	96.21	89.06	76.82	97.00	89.24	77.00	96.35	83.36	63.92	96.34	83.69	64.12
XLMR – CSSL	96.37	91.03	80.60	97.85	91.16	80.43	96.75	83.54	64.32	97.21	84.07	65.38
DiffNeCTIS (Ours)	**98.10**	**94.61**	**81.57**	**98.92**	**94.71**	**82.06**	**97.25**	**83.40**	**65.04**	**98.71**	**86.11**	**68.33**
+ Local Refinement	–	–	–	–	–	–	–	–	–	*96.82*	*83.94*	***74.01***
NeCTIS OOD												
LSTM – CE	93.32	78.94	57.40	95.67	79.96	57.30	93.88	67.96	36.60	95.88	67.26	36.26
LSTM – CSSL	95.97	80.55	59.18	98.18	80.79	59.56	94.10	68.21	37.24	96.24	68.45	38.29
XLMR – CE	95.56	84.24	65.00	95.56	84.45	65.50	93.88	74.26	45.70	95.45	73.54	44.37
XLMR – CSSL	96.46	85.94	65.21	95.57	85.26	65.90	94.24	75.02	45.79	96.08	73.89	44.98
DiffNeCTIS (Ours)	**98.70**	**86.37**	**66.39**	**98.43**	**85.24**	**65.76**	**96.49**	**77.71**	**47.08**	**97.29**	**74.15**	**45.29**
+ Local Refinement	–	–	–	–	–	–	–	–	–	*96.19*	*73.13*	***49.02***

Table 2: Evaluation on the NeCTIS datasets, considering 2 levels of annotations (Coarse: 4 labels, Fine-grained: 56 labels) and 2 settings (with and without context). Best results in **bold**, second-best underlined and Local Refinement results being *italicized*. Results averaged over 3 runs.

and without context. We run instruction fine-tuning and zero-shot experiments with **Gemma-3-4B**, **Gemma-3-12B-IT**, and **Gemma-2-27B-IT** (Team, 2024), and **Qwen-2.5-32B** (Hui et al., 2024); all models are loaded in 4-bit via BitsAndBytes (Dettmers et al., 2023). We further probe constrained decoding, reranking, and beam search on Qwen-2.5-32B (Appendix Table 10). The full prompt template is in Appendix A.4.

Discriminative baselines. DepNeCTI applies a bi-affine attention module (Dozat and Manning, 2017) to encoder representations (LSTM or XLM-R versions) to transform NeCTIS into dependency parsing. We report both the LSTM version with cross-entropy (**LSTM-CE**) and with constrastive loss (**LSTM-CSSL**) (Ray et al., 2024), as well as the XLM-R encoder variants (**XLMR-CE**, **XLMR-CSSL**) (Ray et al., 2024). These represent the earlier state of the art for NeCTIS.

5 Results and Analysis

5.1 Main Results

Table 2 presents in-domain (NeCTIS) and out-of-domain evaluations across coarse and fine-grained settings. In the coarse-grained task, DiffNeCTIS consistently outperforms the strongest discriminative baseline (XLMR+CSSL) regardless of sentential context, yielding improvements of up to +3.55 LSS and +1.63 EM with context (82.06 vs. 80.43). This demonstrates that diffusion-based decoding inherently provides a stronger inductive bias for structural prediction than standard bi-affine dependencies. This advantage amplifies in the more challenging fine-grained setting (56 labels). With context, DiffNeCTIS surpasses XLMR-CE by +2.37 USS, +2.42 LSS, and +4.21 EM, highlighting its superior capacity for joint structure-and-label prediction.

Local Refinement. In the fine-grained with-context configuration, window-restricted attention in the diffusion decoder yields a significant EM gain: 74.01 vs. 68.33 for DiffNeCTIS alone (+5.68 EM), achieving the highest EM among all systems. Local Refinement exhibits a precision-recall trade-off: USS decreases from 98.71 to 96.82 (−1.89) and LSS drops from 86.11 to 83.94 (−2.17), while EM improves substantially from 68.33 to 74.01 (+5.68) as shown in Table 2, the confined receptive field improves label precision at the cost of global structural accuracy. For comparison, bi-affine conditioning during reverse diffusion achieves 74.14 LSS and 71.40 EM, superior to DiffNeCTIS alone but inferior to Local Refinement, while suffering from severely degraded USS (91.79), confirming that cascaded architectures are prone to error propagation.

Out-of-domain generalization. DiffNeCTIS generalizes robustly on NeCTIS OOD. In the coarse setting, it achieves 98.70 USS and 86.37 LSS without context, and 98.43/85.24/65.76

	Coarse						Fine-grained					
	w/o ctx			w/ ctx			w/o ctx			w/ ctx		
Model	USS	LSS	EM	USS	LSS	EM	USS	LSS	EM	USS	LSS	EM
Gemma-3-27B-IT (5-shot inference)	43.43	33.51	0	44.04	33.57	0	42.51	21.38	0	43.32	20.98	0
Gemma-3-27B (IFT)	73.50	**55.50**	31.50	74.23	**56.38**	32.50	72.51	43.60	15.79	72.46	42.98	12.06
Qwen-2.5-32B (5-shot inference)	49.79	38.06	0	50.92	38.24	0	48.35	24.15	0	49.21	23.87	0
Qwen2.5-32B (IFT)	**80.12**	50.67	**31.97**	**81.25**	51.64	**34.16**	**78.92**	**46.57**	**18.23**	**79.18**	**45.82**	**15.92**

Table 3: Evaluation on the NeCTIS dataset using LLMs under two annotation levels (Coarse: 4 labels, Fine-grained: 56 labels) and two settings (with and without context), comparing few-shot inference and instruction fine-tuning (IFT) for large language models.

(USS/LSS/EM) with context, representing gains of +2.76 USS, +5.28 LSS, and +8.46 EM over LSTM-CE. In the fine-grained OOD setting with context, improvements over LSTM-CE reach +1.41 USS, +6.89 LSS, and +9.03 EM.These results indicate that the diffusion model's iterative refinement confers greater domain shift resilience than single-pass discriminative models.

5.2 LLM Evaluation and Failure Analysis

LLM evaluation reveals a clear size-dependent failure pattern. Smaller models (**Gemma-3-4B**, **Gemma-3-12B**) produce no meaningful output, scoring 5-10% on USS and LSS metrics. In the few-shot setting, larger models (**Gemma-2-27B**, **Qwen-2.5-32B**) achieve moderate USS and LSS scores but fail to recover complete compound structures, yielding EM scores of zero across all settings. In contrast, instruction fine-tuning (IFT) substantially improves performance: Gemma-3-27B-IFT and Qwen-2.5-32B-IFT, as shown in Table 3 achieves notable gains across all metrics, including non-zero EM scores. This confirms that broad linguistic pretraining is insufficient for NeCTIS: the task demands structured reasoning and task-specific inductive biases that general-purpose LLMs do not provide as shown in Appendix A.5.

5.3 Ablation Study and Analysis

We ablate key components of DiffNeCTIS on the fine-grained with-context NeCTIS setting. Results are summarized in Table 7.

Self-conditioning. Removing self-conditioning (Strudel et al., 2022) from the diffusion decoder reduces LSS by 1.87 points (86.11 $\rightarrow$ 84.24) and EM by 2.58 points, while USS remains comparable (98.81) as shown in Table 7. This confirms that self-conditioning primarily benefits label prediction quality rather than structural accuracy, consistent with its role in sharpening the denoising signal during iterative refinement.

Denoising steps. Increasing λ from 1 to 50 as shown in Table 7 yields monotonic improvement. Beyond $\lambda = 100$, performance slightly degrades, suggesting that 50 steps strikes an effective balance between refinement quality and inference cost. Inference time scales linearly with λ.

Window size. Varying w reveals in Table 2 a clear tradeoff: smaller windows ($w = 3$) maintain EM but degrade USS more aggressively, while larger windows ($w = 9$) recover USS but reduce the EM gains. The default window size ($w = 5$) provides the best configuration. This shows that window-restricted attention improves fine-grained label precision but limits the receptive field available for structural decisions.

Biaffine conditioning: To inject pairwise relational signals, we augment the BitDiT decoder with bi-affine features computed directly over encoder representations $\mathbf{H}$, bypassing any coarse conditioning. For each span position i, the bi-affine score between head and tail representations is:

$$s_{ij} = \mathbf{h}_i^\top \mathbf{W}_{\text{biaff}}\, \mathbf{h}_j + \mathbf{u}^\top \mathbf{h}_i + \mathbf{v}^\top \mathbf{h}_j \qquad (11)$$

where $\mathbf{W}_{\text{biaff}} \in \mathbb{R}^{d\times d}$, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ are learned parameters. The projected score matrix is added to the conditioning signal at every denoising step, providing $\mathcal{O}(1)$ pairwise sensitivity that complements rather than replaces the $\mathcal{O}(1/n)$ global attention signal. To preserve coarse structural regularization without introducing cascade dependencies, we add an auxiliary coarse classification loss on the encoder output:

$$\mathcal{L} = \mathcal{L}_{\text{diff}} + \mu\, \mathcal{L}_{\text{coarse}} \qquad (12)$$

where μ controls the regularization strength and $\mathcal{L}_{\text{coarse}}$ is a cross-entropy loss over the 4-class coarse taxonomy applied to $\mathbf{H}$; it serves purely

		Diff.			+ Ref.		
N	#	USS	LSS	EM	USS	LSS	EM
2-3	4,569	98.9	85.9	68.9	99.1	86.2	70.0
4-6	1,007	99.3	81.8	48.6	99.4	82.9	49.9
7-9	114	99.6	80.9	28.9	99.8	82.9	34.2
10-12	20	99.8	56.8	30.0	99.8	87.8	40.0
12+	4	99.1	90.3	25.0	100.0	93.0	25.0

Table 4: Performance stratified by compound length (N = number of components) for DiffNeCTIS and + Local Refinement on fine-grained NeCTIS (w/ context). # denotes the number of compounds per bin.

as an inductive bias on the encoder and does not condition the reverse diffusion process. As shown in Table 7, does not recover the structural accuracy of the base model, exhibiting degraded USS and LSS.

5.4 Analysis of the USS/LSS/EM Tradeoff

Breakdown by compound length. Table 4 stratifies results by the number of components N. For short compounds (N = 2-3), which comprise 81% of the test set (4,569 instances), Local Refinement improves EM by +1.1% (68.9% $\rightarrow$ 70.0%) while maintaining comparable USS and LSS. The most pronounced gains emerge on longer compounds: at N = 7-9, EM improves by +5.3% (28.9% $\rightarrow$ 34.2%) with a corresponding +2.0% LSS gain; at N = 10-12, EM improves by +10.0% (30.0% $\rightarrow$ 40.0%). For the longest compounds ($N > 12$), USS reaches 100% and LSS improves by +2.7%.

Error Type	DiffNeCTIS	+ Local Ref.	Δ
Correct	63.15%	64.66%	+1.50
Label Error	28.67%	28.13%	−0.54
Span Error	6.57%	6.23%	−0.33
Missed	1.61%	0.98%	−0.63

Table 5: Error type distribution over 6,014 gold spans. *Correct*: both span and label match; *Label Error*: span correct, label wrong; *Span Error*: boundaries incorrect; *Missed*: no overlapping prediction.

Error type classification. Table 5 categorizes each gold span into four error types: *correct* (both span and label match), *label error* (span boundaries correct, label wrong), *span error* (boundaries incorrect), and *missed* (no overlapping prediction). DiffNeCTIS produces 28.67% label errors and 6.57% span errors, while Local Refinement reduces label errors to 28.13% and span errors to 6.23%, with missed spans dropping from 1.61% to 0.98%. Notably, Local Refinement improves all error categories simultaneously, with no trade-off between label precision and structural recall.

Coarse Type	#	Diff.	+ Ref.	Δ LSS
Tatpurusa	4,980	34.8	34.9	+0.1
Dvandva	647	30.6	34.3	+3.8
Other	221	29.0	30.2	+1.2
Bahuvrīhi	166	33.9	33.8	−0.1

Table 6: Per-type fine-grained LSS breakdown by coarse compound type, computed as span-level accuracy within each coarse class (not micro-averaged F1 as in Table 2).

Per-class analysis. Table 6 breaks down fine-grained LSS by coarse compound type. Contrary to our initial hypothesis that Tatpurusa subtypes would benefit most from local attention (due to their reliance on case-marking distinctions), Dvandva (copulative) compounds show the largest gain (+3.8%), while Tatpurusa improves by only +0.1%.

The near-zero Tatpurusa improvement is an artifact of aggregation: T6 (ṣaṣṭhī-tatpuruṣa), comprising 45% of all Tatpurusa spans, degrades by −2.5% under Local Refinement, offsetting gains in T3 (+5.0%), Bs6 (+3.3%), K1 (+1.8%), and K6 (+4.3%). T6 involves semantically mediated genitive relations requiring broader contextual scope than $w = 5$ provides, whereas Dvandva's strong gain (+3.9% for Di) reflects that copulative coordination is discriminated by adjacent conjunct identity, a purely local signal. This reveals a *type-dependent Granularity Gap*: locally-discriminated types benefit from window restriction while structurally-mediated types require global scope as shown in Table 8.

6 Conclusion

We investigated structured generative approaches for Nested Compound Type Identification in Sanskrit (NeCTIS), reformulating the task as diffusion-based sequence labeling. Our model, DiffNeCTIS, outperforms discriminative bi-affine baselines across both coarse and fine-grained settings. We identified a *Granularity Gap* in global-attention diffusion models and proposed a coarse-to-fine local refinement mechanism that substantially improves exact match accuracy with modest USS and LSS trade-offs. Our LLM evaluation confirms that current large language models fail on NeCTIS by wide margins, underscoring the continued need for task-specific structured architectures in morphologically complex, low-resource settings.

7 Limitations

Our work has several limitations. First, the diffusion-based decoder requires iterative DDIM sampling at inference, making it slower than single-pass discriminative models; we use 50 denoising steps as a speed–quality trade-off, but latency remains higher. Second ,the local refinement module's window-restricted attention improves EM but degrades USS, and LSS on the in-domain test set and OOD generalization, suggesting the two-stage cascade introduces domain sensitivity that warrants further investigation.. Third, our evaluation is limited to Sanskrit due to the absence of comparable multi-component compound datasets in other languages; generalizability to other morphologically rich languages remains untested. Finally, hyperparameter sensitivity particularly the noise schedule, SNR scaling, and window size requires careful tuning, and we did not conduct an exhaustive search over these choices.

References

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Cheung. 2022. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 700–710, Dublin, Ireland. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

M. Coulson. 1976. *Sanskrit: An Introduction to the Classical Language*. Teach yourself books. Hodder and Stoughton.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. *Preprint*, arXiv:1611.01734.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626, Sydney, Australia. Association for Computational Linguistics.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2023. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *Preprint*, arXiv:2210.17432.

Oliver Hellwig. 2010–2021. Dcs - the digital corpus of sanskrit.

Oliver Hellwig. 2016. Improving the morphological analysis of classical Sanskrit. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 142–151, Osaka, Japan. The COLING 2016 Organizing Committee.

Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit word segmentation using character-level recurrent and convolutional neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2754–2763, Brussels, Belgium. Association for Computational Linguistics.

Oliver Hellwig and Wiebke Petersen. 2012. Aṣṭādhyāyī 2.0: Digital research environment for pāṇini's grammar. University of Düsseldorf.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Preprint*, arXiv:2006.11239.

Ziyang Huang, Pengfei Cao, Jun Zhao, and Kang Liu. 2023. Diffusionsl: Sequence labeling via tag diffusion process. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12907–12921.

Gérard Huet. 2009. Sanskrit segmentation. INRIA.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Amrith Krishna, Pavankumar Satuluri, Shubham Sharma, Apurv Kumar, and Pawan Goyal. 2016. Compound type identification in Sanskrit: What roles do the corpus and grammar play? In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 1–10, Osaka, Japan. The COLING 2016 Organizing Committee.

Malhar Kulkarni and Chaitali Dangarikar. 2013. Recent researches in sanskrit computational linguistics.

Anil Kumar and Amba Kulkarni. 2012. An automatic sanskrit compound processing.

Peichao Lai, Jiaxin Gan, Feiyang Ye, Yilei Wang, and Bin Cui. 2025. Improving low-resource sequence labeling with knowledge fusion and contextual label explanations. *Preprint*, arXiv:2501.19093.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.

K. V. Ramkrishnamacharyulu, Amba Kulkarni, Tirumala Kulkarni, and Anil Kumar. 2012. Guidelines for tagging of sanskrit compounds. Final draft for circulation among SHMT consortium members.

Pretam Ray, Jivnesh Sandhan, Amrith Krishna, and Pawan Goyal. 2024. CSSL: Contrastive self-supervised learning for dependency parsing on relatively free word ordered and morphologically rich low resource languages. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8458–8466, Miami, Florida, USA. Association for Computational Linguistics.

Jivnesh Sandhan, Laxmidhar Behera, and Pawan Goyal. 2023a. Systematic investigation of strategies tailored for low-resource settings for low-resource dependency parsing. *Preprint*, arXiv:2201.11374.

Jivnesh Sandhan, Ashish Gupta, Hrishikesh Terdalkar, Tushar Sandhan, Suvendu Samanta, Laxmidhar Behera, and Pawan Goyal. 2022. A novel multi-task learning approach for context-sensitive compound type identification in Sanskrit. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4071–4083, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Jivnesh Sandhan, Yaswanth Narsupalli, Sreevatsa Muppirala, Sriram Krishnan, Pavankumar Satuluri, Amba Kulkarni, and Pawan Goyal. 2023b. Depnecti: Dependency-based nested compound type identification for sanskrit. *arXiv preprint arXiv:2310.09501*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Diffusionner: Boundary diffusion for named entity recognition. *Preprint*, arXiv:2305.13298.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2022. Denoising diffusion implicit models. *Preprint*, arXiv:2010.02502.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. 2022. Self-conditioned embedding diffusion for text generation. *Preprint*, arXiv:2211.04236.

Gemma Team. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. 2013. Parsing morphologically rich languages: Introduction to the special issue. *Computational Linguistics*, 39(1):15–22.

J. C. Wright. 1968. J. f. staal: Word order in sanskrit and universal grammar. (foundations of language supplementary series, vol. 5.) xi, 98 pp. dordrecht: D. reidel publishing co., 1967. guilders 30. *Bulletin of the School of Oriental and African Studies*, 31(1):205–205.

Qiaoxuan Yin, Jianquan Ouyang, and Huanrong Tang. 2025. Exploring iterative refinement for nested named entity recognition with iou-aware denoising diffusion. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, page 3888–3898, New York, NY, USA. Association for Computing Machinery.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

A Appendix

A.1 Why NeCTIS as Dependency Task?

As discussed in (Sandhan et al., 2023b), a number of factors led to the choice to structure NeCTIS as a dependency parsing task. Usually, successive binary combinations that adhere to particular semantic relations are used to produce compounds with more than two components. With a few exceptions, this procedure produces a nested structure of binary compounds that forms a constituency tree. There are difficulties, nevertheless, with approaching this as a constituency parsing problem. Compounds' layered structure follows a semantic structure based on component relations rather than a syntactic framework. Annotating the types as directed relations allows us to describe the same spans as dependency structures if we replace the intermediary nodes with semantic relations. Furthermore, headwords in constituency spans can be recognized by their respective kinds even though they are not marked explicitly. Dependency structures, on the other hand, make it possible to identify the headword from labels that are

applied to it within the compound. Notably, dependency structures can be mutually converted with their corresponding spans and accurately contain consistency information. In summary, dependency relations, which can be expressed as directed labels inside the dependency parse structure, are similar to the semantic interactions between compound components. The semantic relations are succinctly represented by this method without the need for intermediate nodes. Finally, it makes it possible to identify compound types and structure or constituency span at the same time.

Algorithm 1 Training (forward process).

Require: Noise schedule $\{\bar{\alpha}_1 \cdots \bar{\alpha}_T\}$; word encoder $\mathcal{E}_\theta : \mathcal{V}^n \to \mathbb{R}^{n\times d}$; decoder $\mathcal{D}_\phi : (\mathbb{R}^{n\times m}, \mathbb{N}, \mathbb{R}^{n\times d}) \to \mathbb{R}^{n\times m}$

for all (X, ℓ) in dataset **do**
  $\mathbf{w} = \mathcal{E}_\theta(X)$ {word embeddings}
  Initial signal: $\mathbf{x}_0 = \texttt{tag2bit}^*(\ell)$
  $t \sim \text{Unif}(1, T)$ {sample timestep}
  $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {Gaussian noise}
  Latent variable: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\mathbf{e}$
  Gradient descent step on:
    $\nabla_{\theta,\phi} = \|\mathbf{e} - \mathcal{D}_\phi(\mathbf{x}_t, t, \mathbf{w})\|^2$
end for

Algorithm 2 Inference with Local Refinement.

Require: Sentence $X = (x_1 \cdots x_M) \in \mathcal{V}^n$, skip steps s, window size w.

Ensure: Fine-grained tag sequence $\tilde{\ell}^{\text{fine}}$.

$\mathbf{H} = \mathcal{E}_\theta(X)$ {shared encoder representations}

// - Stage 1: Global Diffusion (coarse labels) -
$\mathbf{x}_T^{(1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {sample noise for coarse stage}
$t \leftarrow T$
while $t > 0$ **do**
  $\hat{\mathbf{x}}_0^{(1)} = \mathcal{D}_\phi^{\text{global}}(\mathbf{x}_t^{(1)}, t, \mathbf{H})$ {global self-attention}
  $k = \max(t - s,\ 0)$
  $\mathbf{x}_k^{(1)} = \frac{\sqrt{\bar{\alpha}_k}}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t^{(1)} - \sqrt{1-\bar{\alpha}_t}\,\hat{\mathbf{x}}_0^{(1)}\right) + \sqrt{1-\bar{\alpha}_k}\,\hat{\mathbf{x}}_0^{(1)}$ {DDIM step}
  $t \leftarrow k$
end while
$\hat{y}^{\text{coarse}} = \texttt{bit2tag}^*(\mathbf{x}_0^{(1)})$ {decode coarse labels}
// - Stage 2: Local Refinement (fine-grained labels) -
$\mathbf{c} = h_{\text{time}}(t) + h_{\text{coarse}}(\hat{y}^{\text{coarse}}) + \mathbf{H}$ {conditioning signal}
$\mathbf{x}_T^{(2)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {sample noise for fine stage}
$t \leftarrow T$
while $t > 0$ **do**
  $\hat{\mathbf{x}}_0^{(2)} = \mathcal{D}_\psi^{\text{local}}(\mathbf{x}_t^{(2)}, t, \mathbf{c})$ {window attention, $|W(i)| = w$}
  $k = \max(t - s,\ 0)$
  $\mathbf{x}_k^{(2)} = \frac{\sqrt{\bar{\alpha}_k}}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t^{(2)} - \sqrt{1-\bar{\alpha}_t}\,\hat{\mathbf{x}}_0^{(2)}\right) + \sqrt{1-\bar{\alpha}_k}\,\hat{\mathbf{x}}_0^{(2)}$ {DDIM step}
  $t \leftarrow k$
end while
$\tilde{\ell}^{\text{fine}} = \texttt{bit2tag}^*(\mathbf{x}_0^{(2)})$ {decode fine-grained labels}
return $\tilde{\ell}^{\text{fine}}$

A.2 Denoising Network: Bit Diffusion Transformer

The Bit Diffusion Transformer (BitDiT) serves as the denoising network in our framework. It consists of a BERT-style encoder that processes the input sentence and a decoder that generates clean tags in parallel. The decoder employs adaptive layer normalization (AdaLN) to incorporate conditional signals (timestep and sentence encodings), replacing traditional cross-attention mechanisms.

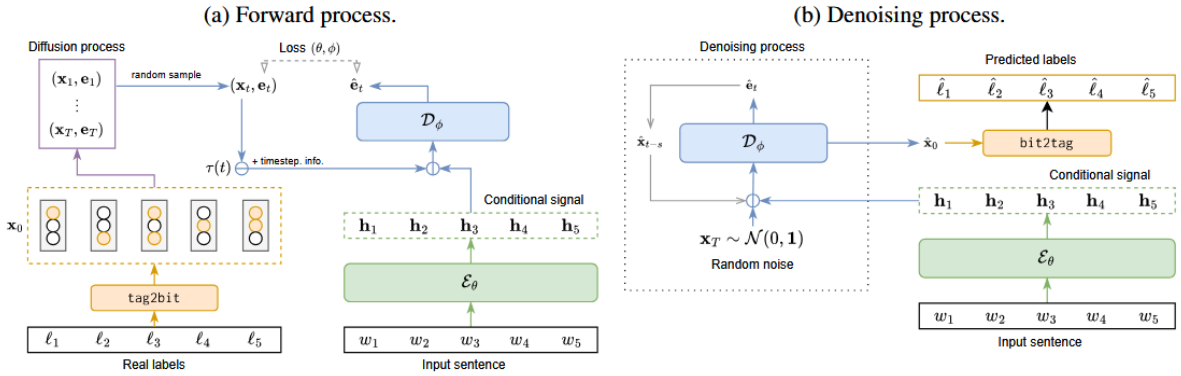

Figure 3: In the diffusion sequence labeling architecture, the forward and denoising steps are distinct yet complementary. The symbol $\ominus$ denotes the concatenation operator, and an open arrow represents loss propagation. During the forward process, the encoder E_θ embeds the input sentence to serve as the conditional signal. The real labels are transformed into bits and fed into the diffusion process, where the latent variable x_t is computed from the sampled noise e_t and concatenated with time embeddings $\tau(t)$ and the conditional signal. Subsequently, the decoder D_ϕ learns to extract the noise that was added to x_t, with all parameters optimized using the MSE loss between the real and predicted noise. In the denoising process (Figure 3b), the conditional signal is computed once via E_θ, and an initial signal x_T is sampled from Gaussian noise. Iteratively, D_ϕ removes noise from the input, utilizing the conditional signal to estimate the previous latent state $\hat{x}_{t-s}$ until the fully denoised state $\hat{x}_0$ is reached. Finally, $\hat{x}_0$ is fed back to the BT module to recover the sequence of predicted labels.

Configuration	USS	LSS	EM
DiffNeCTIS (full)	**98.71**	**86.11**	**68.33**
− self-conditioning	98.81	84.24	65.75
+ biaffine cond.	93.79	80.14	71.40
Denoising steps λ			
$\lambda = 1$	90.18	64.12	29.32
$\lambda = 10$	93.79	73.48	45.21
$\lambda = 50$ (default)	**98.71**	**86.11**	**68.33**
$\lambda = 100$	97.79	84.39	67.92
Window size w (Local Refinement)			
$w = 3$	95.16	81.95	72.16
$w = 5$ (default)	96.82	83.94	**74.01**
$w = 9$	**97.37**	**84.13**	71.96

Table 7: Ablation study on fine-grained NeCTIS (w/ context). λ: denoising steps; w: window size.

This design choice accelerates convergence and improves final performance.

The decoder comprises four components: (1) a **Condition Embedder** that fuses timestep and sentence representations, (2) a **Pre Layer** (MLP) that projects bit representations to higher dimensions, (3) a **Generator** with N Transformer blocks using AdaLN, and (4) a **Post Layer** that normalizes and projects back to the original dimension. Learned scale factors control the amount of newly learned information before each residual connection. The overall generation process is:

$$\tilde{\mathbf{x}}_0 = \text{Post}(\text{Generator}(\text{Pre}(\mathbf{x}_t), \mathbf{c}_t)) \quad (13)$$

where $\mathbf{c}_t$ represents the conditional signal.

The Transformer block architecture used in the BitDiT decoder is illustrated in Figure 4. Each block processes noisy tag bits $\mathbf{x}_t$ through two sequential sub-blocks:

1. **First sub-block**: Adaptive Layer Normalization (ALN) → Multi-Head Self-Attention (MHSA) → Scaled skip-connection

2. **Second sub-block**: ALN → Feed-Forward Network (FFN) → Scaled skip-connection

The computation proceeds as:

$$\mathbf{x}_t = \alpha_1 \times \text{MHSA}(\text{ALN}_1(\mathbf{x}_t), \mathbf{c}_t) + \mathbf{x}_t \quad (14)$$
$$\mathbf{x}_t = \alpha_2 \times \text{FFN}(\text{ALN}_2(\mathbf{x}_t), \mathbf{c}_t) + \mathbf{x}_t \quad (15)$$
$$\alpha_{1,2} = \text{Linear}_{1,2}(\mathbf{c}_t) \quad (16)$$

where α_1 and α_2 are learned scale factors conditioned on $\mathbf{c}_t$.

The Adaptive Layer Normalization learns scale γ and shift β parameters based on the conditional embedding $\mathbf{c}_t$:

Type	Coarse	#	DiffNeCTIS	+ Local Ref.	Δ LSS
T6	Tat.	2,230	40.5	38.0	−2.5
K1	Tat.	564	32.7	34.5	+1.8
Di	Dvan.	605	31.7	35.6	+3.9
Bs6	Tat.	407	30.0	33.3	+3.3
Tn	Tat.	338	36.0	36.8	+0.9
K7	Tat.	270	36.6	37.5	+0.9
T3	Tat.	247	27.5	32.4	+5.0
U	Other	141	30.5	31.2	+0.7
Tp	Tat.	129	39.6	37.9	−1.7
T7	Tat.	106	15.8	17.3	+1.5
BvS	Bahu.	81	37.6	39.0	+1.4
Bv	Bahu.	55	36.0	31.6	−4.4
K6	Tat.	50	16.3	20.6	+4.3
A1	Other	44	29.4	33.0	+3.6
Ds	Dvan.	41	13.8	15.8	+1.9

Table 8: Fine-grained LSS for the 15 most frequent sub-types. T3 shows the largest gain (+5.0%), while T6, the most frequent type (37% of all spans) degrades by −2.5%, explaining the near-zero aggregate Tatpurusa improvement in Table 6.

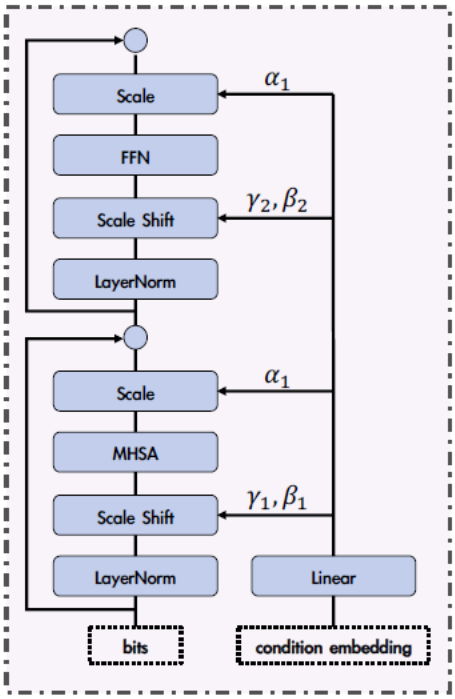

Figure 4: Transformer block details of the BitDiT Decoder. MHSA represents Multi-Head Self-Attention and FFN represents Feedforward Network. β and γ are the learned affine parameters for adaptive layer normalization. α is the scale factor before each skip connection.

$$\mathrm{ALN}(\mathbf{x}_t, \mathbf{c}_t) = \gamma \times \mathrm{LN}(\mathbf{x}_t) + \beta \tag{17}$$

$$\mathrm{LN}(\mathbf{x}_t) = \frac{\mathbf{x}_t - \mathbb{E}(\mathbf{x}_t)}{\sqrt{\mathrm{Var}(\mathbf{x}_t) + \epsilon}} \tag{18}$$

$$\gamma = \mathrm{Linear}_3(\mathbf{c}_t) \tag{19}$$

$$\beta = \mathrm{Linear}_4(\mathbf{c}_t) \tag{20}$$

where ϵ ensures numerical stability and prevents division-by-zero errors.

This architecture enables the model to dynamically adjust normalization parameters and residual contributions based on both the current timestep and input context, leading to more effective denoising compared to standard cross-attention mechanisms.

A.3 Diffusion VS Structured Prediction

We compare is Table 9 the properties if various axis as to why DiffNeCTIS makes sense even though it incures more cost to train and infer.

A.4 Prompt For Large Language Models

The full prompt used for inference is shown in Figure 5.

A.5 LLM Failure analysis

A qualitative analysis comparing the gold spans of the NeCTIS dataset in the without context coarse classification setting, with the model predictions, reveals several consistent error patterns. The most common issue is that the model shows *Tatpuruṣa overgeneralization*, often predicting Tatpuruṣa even when the gold label corresponds to another compound type (e.g., for input `piśāca ādi ravaḥ` the model predicts `(0,3 Tatpuruṣa)` instead of `(0,3 Bahuvrihi)`. Similarly, the model frequently misclassifies Dvandva compounds as Tatpuruṣa; e.g., the input `mṛga-vāyasa-mūṣikāḥ` is predicted as `(0,3 Tatpuruṣa)` instead of the gold compound span `(0,3 Dvandva)`. Occasional *Avyayībhāva confusions* are observed, e.g., for input `vacana`

Axis	CRF	CKY (TreeCRF)	DiffNeCTIS
Factorization	Pairwise transitions	Span/rule scores	**None – full-sequence joint denoising**
Label-space cost	$O(K^2)$ transitions	$O(K \cdot G)$ rules	$O(K)$ **(bit-encoded)**
Global consistency	Approximate (local factors)	Exact under PCFG	**Implicit via iterative refinement**
Inference	Forward–backward / Viterbi	$O(n^3K^3)$	$O(\Lambda \cdot n^2)$ **parallel**

Table 9: Comparison of structured prediction properties across CRF, CKY (TreeCRF), and DiffNeCTIS.

sama kālam is predicted as (0,3 Avyayibhava) instead of (0,3 Tatpuruṣa), suggesting misinterpretation of compounds. Another class of errors involves *boundary and span segmentation*, where the predicted span structure differs from the gold annotation; for example, for input pada artheṣu vākya artha avagama śakti ādhāna śaktiḥ the model predicts (0,2 Tatpuruṣa | 2,4 Tatpuruṣa | 6,8 Tatpuruṣa) instead of (0,2 Tatpuruṣa | 2,8 Tatpuruṣa), introducing an additional split. Finally, the model struggles with *long nested compounds* (e.g., for the input <mantra auṣadhi maṇi maṅgala bali upahāra homa niyama prāyaścitta upavāsa svastyayana praṇipāta gamana ādi is decomposed into several Tatpuruṣa spans (0,2 Tatpuruṣa|3,5 Tatpuruṣa|8,10 Tatpuruṣa|10,12 Tatpuruṣa|11,13 Tatpuruṣa) instead of (0,14 Dvandva), indicating difficulty in modelling complex enumerative structures.

Configuration	USS	LSS	EM
Qwen-2.5-32B (Fine-grained, w/ context)			
IFT (baseline)	79.18	45.82	15.92
+ Beam Search	80.50	47.22	17.57
+ CD	83.01	50.48	22.15
+ Reranking (k=5, no CD)	81.98	48.61	22.76
+ CD + Reranking + Beam	**84.09**	**52.46**	**33.70**

Table 10: Ablation of decoding strategies for Qwen-2.5-32B on the fine-grained NeCTIS setting (w/ context). CD: Contrastive Decoding; Reranking with $k = 5$ candidates. Best results in **bold**.

A.6 Hyperparameters for Instruction FineTuning LLMs

We evaluate our approach using the instruction-tuned large language model *Gemma-3-4B, Gemma-3-12B-IT, and Gemma-2-27B-IT from the Gemma model family, as well as Qwen-2.5-32B* implemented through the Hugging Face Transformers framework. The model is deployed using 4-bit NF4 quantization via BitsAndBytes with bfloat16 computation to enable memory-efficient inference. Generation is performed with a maximum input length of 2048 tokens and up to 400 generated tokens using deterministic decoding (do_sample=False). Inference is executed with a batch size of 8 and left-padding for efficient decoding. Evaluation is conducted on test sentences consisting of segmented Sanskrit tokens annotated with compound spans. Model outputs are converted to the NeCTIS evaluation format and evaluated using Unlabeled Segmentation Score (USS) and Labeled Segmentation Score (LSS), reported as F1 scores over predicted and gold spans. We additionally report Exact Match (EM), measuring the proportion of predicted compound structures that exactly match the gold nested structures.

Parameter	Value	Parameter	Value
Model Architecture			
Backbone	`xlm-roberta-large`	Network Architecture	`transformer`
`dim_model`	1024	`dim_time`	512
Depth	12	Objective	`pred_x0`
Time Steps	100	Sampling Steps	50
DDIM Sampling η	1.0	Noise Schedule	cosine
Loss Type	L2	Self-Condition	True
SNR Scale	1.0	Add LSTM	False
Freeze BERT	False	Ensemble	False
Max Length	256	Num Classes	56
Training			
Batch Size	16	Accumulation Steps	1
Max Epochs	200	Max Steps	500,000
Optimizer	AdamW	LR Scheduler	cosine
lr_{BERT}	2×10^{-5}	lr_{other}	5×10^{-4}
Weight Decay	0.01	Max Grad Norm	1.0
Warmup Steps	1,000	Warmup Ratio	0.01
Num Workers	4	GPUs	1
Data			
Dataset	`necti_finegrain`	Granularity	Finegrain

Table 11: Hyperparameters and Configuration for NeCTI Finegrained Nested Compound Identification (XLM-RoBERTa + DiffusionSL)

You are an expert Sanskrit linguist specializing in compound word (samāsa) analysis. Your task is to identify ALL compounds in a segmented Sanskrit sentence, including nested compounds.

COMPOUND TYPES:

1. **Tatpurusha** : Determinative compound where first element modifies second - Test: Can be paraphrased as X-sya Y (Y of X) - Example: rāja-putra → rājñaḥ putraḥ (king's son)
2. **Bahuvrihi** : Possessive compound referring to an external entity - Test: Can be paraphrased as yasya X asti (one who has X) - Example: su-mukha → yasya sundaram mukham asti (one who has a beautiful face)
3. **Dvandva** : Copulative compound coordinating elements with "and" - Test: Can be paraphrased as X ca Y ca (X and Y) - Example: rāma-kṛṣṇa → rāmaḥ ca kṛṣṇaḥ ca (Rama and Krishna)
4. **Avyayibhava** : Adverbial compound with indeclinable prefix - Forms an adverb - Example: yathā-śakti → according to ability

CRITICAL RULES:

1. **Indexing**: Use 0-based indexing. End index is EXCLUSIVE. - [0:2] means words at positions 0 and 1 only - [2:5] means words at positions 2, 3, and 4
2. **Nested Compounds**: When compounds contain other compounds: - Identify ALL levels (inner and outer) - List INNER compounds BEFORE outer compounds - Example: If words 1-3 form a compound, and words 0-3 form another, list [1:3] first, then [0:3]
3. **Output Format**: Return ONLY valid JSON with all compounds

—

EXAMPLE 1: Complex Mixed (Tatpurusha & Bahuvrihi)

Sentence: "pAda raja upamAH giri nadI vega upamam"
Word indices: 0:pAda 1:raja 2:upamAH 3:giri 4:nadI 5:vega 6:upamam
Total words: 7
Analysis:
Words 0–2 (pAda raja) form Tatpurusha: "dust of feet". Words 0–3 (pAda raja upamAH) form Bahuvrihi: "one who is like the dust of feet". Words 3–5 (giri nadI) form Tatpurusha: "mountain river". Words 3–6 (giri nadI vega) form Tatpurusha: "speed of mountain river". Words 3–7 (giri nadI vega upamam) form Bahuvrihi: "one representing the speed of a mountain river".
Output JSON:
{ "compounds": [{ "start": 0, "end": 2, "type": "Tatpurusha", "text": "pAda raja" }, { "start": 0, "end": 3, "type": "Bahuvrihi", "text": "pAda raja upamAH" }, { "start": 3, "end": 5, "type": "Tatpurusha", "text": "giri nadI" }, { "start": 3, "end": 6, "type": "Tatpurusha", "text": "giri nadI vega" }, { "start": 3, "end": 7, "type": "Bahuvrihi", "text": "giri nadI vega upamam" }] }

—

EXAMPLE 2: Dvandva and Tatpurusha

Sentence: "SIta Atapa varzAByaH surakzA arTaM vetra kawaH"
Word indices: 0:SIta 1:Atapa 2:varzAByaH 3:surakzA 4:arTaM 5:vetra 6:kawaH
Total words: 7
Analysis:
Words 0–3 (SIta Atapa varzAByaH) form Dvandva: "cold, heat, and rain". Words 3–5 (surakzA arTaM) form Tatpurusha: "for the purpose of protection". Words 5–7 (vetra kawaH) form Tatpurusha: "mat made of cane".

```
Output JSON:

\{ "compounds": [
\{ "start": 0, "end": 3, "type": "Dvandva", "text": "SIta Atapa varzAByaH" \},
\{ "start": 3, "end": 5, "type": "Tatpurusha", "text": "surakzA arTaM" \},
\{ "start": 5, "end": 7, "type": "Tatpurusha", "text": "vetra kawaH" \}
] \}
---
EXAMPLE 3: Bahuvrihi with Dvandva List
Sentence: "sa sarzapaM tumburu DAnya vanyaM"
Word indices: 0:sa 1:sarzapaM 2:tumburu 3:DAnya 4:vanyaM
Total words: 5
Analysis:
Words 0-2 (sa sarzapaM) form Bahuvrihi: "accompanied by mustard" (Saha-Bahuvrihi).
Words 2-5 (tumburu DAnya vanyaM) form Dvandva: "tumburu, grain, and wild-grain".
Output JSON:
{ "compounds": [
{ "start": 0, "end": 2, "type": "Bahuvrihi", "text": "sa sarzapaM" },
{ "start": 2, "end": 5, "type": "Dvandva", "text": "tumburu DAnya vanyaM" }
] }
----
EXAMPLE 4:
Sentence: "tyAga saMnyAsa vikalpe yaTA darSite Barata sattama"
Word indices: 0:tyAga 1:saMnyAsa 2:vikalpe 3:yaTA 4:darSite 5:Barata 6:sattama
Total words: 7
Analysis:
Words 0-2 (tyAga saMnyAsa) form Dvandva: "renunciation and resignation".
Words 0-3 (tyAga saMnyAsa vikalpe) form Tatpurusha: "in the option of renunciation-
resignation".
Words 3-5 (yaTA darSite) form Avyayibhava: "as shown".
Words 5-7 (Barata sattama) form Tatpurusha: "best of Bharatas".
Output JSON:
{ "compounds": [
{ "start": 0, "end": 2, "type": "Dvandva", "text": "tyAga saMnyAsa" },
{ "start": 0, "end": 3, "type": "Tatpurusha", "text": "tyAga saMnyAsa vikalpe" },
{ "start": 3, "end": 5, "type": "Avyayibhava", "text": "yaTA darSite" },
{ "start": 5, "end": 7, "type": "Tatpurusha", "text": "Barata sattama" }
] }
---
EXAMPLE 5:
Sentence: "A jIvana kAlam koza pfzWAni"
Word indices: 0:A 1:jIvana 2:kAlam 3:koza 4:pfzWAni
Total words: 5
Analysis:
Words 0-2 (A jIvana) form Avyayibhava: "throughout life / until life".
Words 0-3 (A jIvana kAlam) form Tatpurusha: "time of whole life".
Words 3-5 (koza pfzWAni) form Tatpurusha: "dictionary pages".
```

```
Output JSON:  "compounds": [  "start": 0, "end": 2, "type": "Avyayibhava",
"text": "A jIvana" ,  "start": 0, "end": 3, "type": "Tatpurusha", "text": "A
jIvana kAlam" ,  "start": 3, "end": 5, "type": "Tatpurusha", "text": "koza
pfzWAni"  ]
—
NOW ANALYZE THE FOLLOWING SENTENCE:
Sentence: "<SEGMENTED_SENTENCE>"
Word indices: <WORD_INDICES>
Total words: <NUM_WORDS>
TASK: Carefully identify ALL compounds (including nested ones).  List inner
compounds before outer. Return ONLY the JSON output.
Output JSON:
```

Figure 5: LLM inference prompt used for Sanskrit compound identification on the NeCTIS dataset in the without context coarse classification setting.