Breaking the Script Barrier: Enabling Automatic Alignment for PoS-based ASR Error Analysis in Non-Latin Scripts

Anonymous ACL submission

Abstract

Automatic Speech Recognition (ASR) systems are commonly evaluated using aggregate metrics such as Word Error Rate (WER), which do not capture the linguistic structure of errors. Fine-grained analysis, such as Part-of-Speech (PoS)-wise error characterization, requires accurate alignment between ASR hypotheses and reference transcriptions. However, existing alignment tools are often unreliable for languages written in non-Latin scripts. In this work, we address this gap by proposing a robust, automated, language-agnostic alignment mechanism applicable across ASR architectures and across languages written in both Latin and non-Latin scripts. This enables consistent alignment of hypotheses, references, and evaluation sequences, forming the basis for downstream linguistic analysis. Building on this, we employ standard PoS taggers to perform scalable and reproducible PoS-wise error analysis. Notably, we perform alignment and downstream ASR error analysis across three major segmented writing systems, namely, Abugida (Tamil, Hindi, Kannada), Alphabetic (English, Russian, Greek), and Abjad (Arabic). We further demonstrate how such error information can be leveraged during ASR training to improve metrics such as WER.

1 Introduction

Automatic Speech Recognition (ASR) has become a widely deployed technology, with applications ranging from voice assistants to large-scale transcription systems. Word Error Rate (WER) remains the predominant evaluation metric (von Neumann et al., 2025), along with alternatives such as Match Error Rate (MER), Word Information Lost (WIL) (Morris et al., 2004), Human Perceived Accuracy (HPA) (Mishra et al., 2011), and weighted variants of WER (Le et al., 2016). However, these metrics provide only a coarse-grained view of performance and do not capture the underlying linguistic structure of errors. In particular, they fail to reveal the types of mistakes made in terms of grammatical categories. By incorporating Part-of-Speech (PoS)-wise analysis of errors, we can obtain a deeper understanding of how ASR systems handle different grammatical categories.

Performing automated PoS-wise error analysis over substitution (S), insertion (I), and deletion (D) errors requires aligning the ASR hypothesis (HYP), the reference transcription (REF), and an evaluation string (EVAL) that encodes the error types for each utterance. In standard ASR pipelines, tools such as `sclite` (National Institute of Standards and Technology (NIST), 2000) are used to generate alignment-based evaluation reports. While `sclite` provides reliable alignments for English and other Latin-script languages, its alignment quality significantly degrades for languages written in non-Latin scripts, hence making downstream PoS-based error analysis unreliable. Specifically, for languages such as Tamil, Hindi, Kannada, Arabic, Russian and Greek, we observed that the alignment between the REF, HYP, and EVAL fields was often corrupted due to non-uniform character widths and complex characters, which led to inconsistent spacing. In contrast, languages written in the Latin script, such as English, Spanish, and German, exhibited more uniform spacing, making the default `sclite` alignment sufficiently reliable. We require an alignment algorithm that is able to adapt to non-uniformity in spacing and complexity of characters. Hence we propose a robust alignment mechanism that enables accurate PoS-wise error analysis across languages and scripts.

The Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) was originally developed to align sequences of amino acids to identify similarities and infer evolutionary relationships. It has also been applied to DNA sequence analysis, such as aligning coronavirus genomes to identify mutations (Isa Irawan et al., 2019). While widely

used in bioinformatics, its application to other domains has been relatively limited. More recently, (Bhogale et al., 2023) employed the Needleman-Wunsch algorithm to align audio and text sequences for constructing the Shrutilipi dataset.

Motivated by its effectiveness in sequence alignment, we propose a character-spacing-aware modified variant of the Needleman–Wunsch algorithm to align predicted and reference text sequences from ASR outputs, along with their corresponding evaluation strings. Once the strings are aligned, a PoS tagger is used in order to classify the errors by their parts-of-speech. Note that in standard PoS tagging, only the reference text was assigned PoS labels; therefore, alignment was not required at this stage, since PoS tags are not defined over ASR error strings but over the tokenized reference sequences. However, for PoS-wise error analysis, alignment is crucial and without it one cannot proceed further. We employ the spaCy [1], Stanza (Qi et al., 2020), AI4Bharat (Kakwani et al., 2020), and CAMeL (Obeid et al., 2020) PoS taggers in this work.

Based on our PoS-wise error analysis, we integrate this information into the architecture of the vanilla transformer (Vaswani et al., 2017), obtaining improvement in WER along with grammar-wise improvements. We observe that even assigning higher weights to tokens belonging to parts of speech that are most prone to errors during the computation of decoder cross-attention, reduces the number of errors in these susceptible tokens, thereby ultimately reducing the WER.

To the best of our knowledge, this work is the first to (1) provide an automated alignment mechanism to align the outputs of ASR models applicable across languages and architectures based on (a) modified variant of Needleman-Wunsch algorithm, (b) adaptive character spacing, and (c) post-processing modification of evaluation tags, (2) employ PoS taggers once alignment is done, for the automatic inspection of ASR errors, thereby minimizing manual intervention and enabling scalable, reproducible PoS-wise error analysis, and (3) provide an example of how PoS-wise error information can be utilized in training an ASR model. We show alignment result for the languages (a) Tamil, (b) Hindi, (c) Kannada, (d) Russian, (e) Greek, (f) Arabic, and (g) English. Note that all the above languages have widely different written scripts. According to (Daniels and Bright, 1996), the segmentable languages of the world can be classified into the three writing systems - Abugida, Alphabetic and Abjad. Languages such as Tamil, Hindi and Kannada, where consonant symbols inherently carry vowels, belong to the Abugida writing system. Languages such as Russian, English, and Greek, where vowels and consonants are represented as independent symbols, belong to the Alphabetic writing system. Arabic, where writing is primarily consonant-based and vowels are often omitted in standard text, belongs to the Abjad writing system. The alignment by `sclite` works for languages which have a Latin scipt and belong to the Alphabetic writing system, while our algorithm works for both Latin and non-Latin script languages, breaking the script barrier. We provide details of the alignment mechanism in Section 3, PoS-wise error analysis across languages in Section 4 and results on PoS-aware transformer in Section 5.

[1] https://spacy.io/api/tagger

2 Related Work

Error analysis in ASR Understanding ASR errors has been explored across multiple languages and settings. In (Papadopoulou et al., 2021), transcript quality is evaluated based on human post-editing effort, while (Wirth and Peinl, 2022) analyzes German ASR outputs by categorizing errors into context-breaking, phonetic confusions, and annotation inconsistencies across pretrained models. Similarly, (Chowdhury et al., 2024) studies Bengali ASR transcripts and shows that readability is not always reflected in WER, with errors often arising from speaker variability and unclear word boundaries. However, these approaches rely heavily on manual inspection, limiting scalability and reproducibility. In contrast, our work performs automated, linguistically grounded error analysis using PoS information. More recently, (Parsons et al., 2025) proposed a multi-tiered alignment framework combining character and word-level alignment with speech-production-based similarity metrics and compound reconciliation for one-to-many and many-to-one mappings, evaluated mainly on European languages with Latin scripts. Unlike this approach, our algorithm does not use phonetic or articulatory information, but instead leverages character-width properties and generalizes across diverse segmented writing systems including Alphabetic, Abugida, and Abjad.

PoS integration in model architectures Efforts to

incorporate linguistic information into model architectures have mainly focused on text-based tasks, where inputs are inherently aligned. For instance, the PL-Transformer (Shi et al., 2023) integrates PoS embeddings for text classification using well-formed textual inputs without requiring sequence alignment. In contrast, ASR outputs require explicit alignment between predicted and reference sequences before any meaningful linguistic analysis can be performed.

Similarly, prior works in machine translation and sequence modeling leverage PoS information in settings where alignment is already available. In (James and Krishnamurthy, 2025), word alignment for text-to-text machine translation is improved using PoS-tagged data for Dravidian languages. In (Hlaing et al., 2022), concatenating PoS embeddings with words improves English–Thai translation, while (Wang et al., 2021) uses PoS tags of preceding words to enhance next-word prediction in video captioning. However, these methods operate on text with implicit one-to-one alignment between tokens and their PoS annotations.

3 Existing alignment and proposed approach

In this section, we first briefly discuss the default `sclite` based text alignment and its limitation, followed by our proposed character-spacing aware modified Needleman-Wunsch alignment algorithm. We use the SPRING_INX dataset for Tamil, Hindi and Kannada languages[2](R et al., 2023) and, Common Voice Russian, Greek, Arabic datasets (Ardila et al., 2020) for experimentation. For all results illustrated in this Section, we use the transformer model with 12 encoder and 6 decoder blocks for Tamil and Whisper for Russian and Arabic datasets [3]. For PoS tagging, we use Stanza integrated with spaCy for Tamil and Hindi, spaCy for Russian, AI4Bharat for Kannada, CAMeL for Arabic and gr-nlp-toolkit for Greek[4]. More details about the PoS taggers are provided in Appendix A.

3.1 Default text alignment

Text alignment is done by using `sclite` which is a tool for scoring and evaluating ASR systems, developed as part of the NIST SCTK (National Institute of Standards and Technology Scoring Toolkit). It first matches the unique utterance ID from 'HYP' and 'REF' to create REF-HYP pairs. Next, dynamic programming is used to minimize the Levenshtein distance (Levenshtein, 1966) between REF and HYP. The distance metric assigns fixed, non-adaptive costs of 0, 4, 3, and 3 to correct matches, substitutions, insertions, and deletions respectively (National Institute of Standards and Technology (NIST), 2000). The final alignment is obtained by globally minimizing the cumulative edit cost over all possible alignments using the `sclite` framework. Note that `sclite` is very popular since it works well most European languages, such as English, Spanish, German, etc., a lot of which have a Latin written script. However, languages which have a non-Latin written script, such as Tamil, Hindi, Russian, Greek, Arabic, etc., suffer from significant alignment issues. For the sample ASR output shown in Table 1, the ASR model has made one insertion in the second position and one substitution in the third position. Hence, in Eval, `sclite` aligns 'I' to the first character of the second word, and an 'S' with the first character of the third word. When an insertion occurs, i.e., a word is not present in REF but is present in HYP, `sclite` inserts the character '*' in REF, repeating it for as many characters as the inserted word in HYP, aligned to that word. Similarly, when a deletion occurs, i.e., a word is present in REF but not in HYP, `sclite` inserts '*' in HYP corresponding to that word in REF. In this example, the second word in HYP was inserted, and hence in REF, five '*' characters are inserted corresponding to the 5-character word inserted in HYP. Our objective is to determine the PoS associated with each ASR error. For substitution and deletion errors, we assign the PoS of the REF word, while for insertion errors, we assign the PoS of the inserted HYP word. Thus, if a REF proper noun is recognized as a verb in HYP, the

REF	humpy ***** dumpy fell downstairs
HYP	humpy don't be fell downstairs
Eval	I S

Table 1: Example illustrating S and I errors in alignment produced by `sclite` and our proposed algorithm (both produce perfect alignment)

ref_word	hyp_word	error	PoS
-	don't	I	verb
dumpy	be	S	propn

Table 2: PoS tagging of errors for example in Table 1.

[2]https://asr.iitm.ac.in/dataset
[3]Transformer(Tamil),Whisper(Russian),Whisper(Arabic)
[4]Tamil/Hindi, Russian, Kannada, Arabic, Greek

Alignment using sclite	
REF	**************** அவளுக்கொரு பட்டு பாவாடை தைக்கணும்ன்னு சொல்லி, ஒரு யோசனை இருக்கு.
HYP	அதனால, அவளுக்கு ஒரு பட்பாவோட தைக்கணும்னு சொல்லி, ஒரு யோசனை இருக்கு.
Eval	I S S S S
Alignment using proposed algorithm	
REF	********* அவளுக்கொரு பட்டு பாவாடை தைக்கணும்ன்னு சொல்லி, ஒரு யோசனை இருக்கு.
HYP	அதனால, அவளுக்கு ஒரு பட்பாவோட தைக்கணும்னு சொல்லி, ஒரு யோசனை இருக்கு.
Eval	I S S S S

ref_word	hyp_word	tag	PoS
-	அதனால,	I	adv
**************	அவளுக்கு	S	pron
அவளுக்கொரு	ஒரு	S	pron
பட்டு	பட்பாவோட	S	noun
பாவாடை	தைக்கணும்னு	S	noun

(a) PoS tags after sclite alignment

ref_word	hyp_word	tag	PoS
***************	அதனால,	I	adv
அவளுக்கொரு	அவளுக்கு	S	pron
பட்டு	ஒரு	S	noun
பாவாடை	பட்பாவோட	S	noun
தைக்கணும்ன்னு	தைக்கணும்னு	S	verb

(b) PoS tags after proposed alignment

Table 3: Comparison between `sclite` and proposed alignment for a Tamil ASR example ("avaḷukkoru paṭṭu pāvāṭai taikkaṇumṉṉu colli, oru yōcaṉai irukku.": 'so, there is an idea to stitch her a silk skirt')

substitution is categorized under "proper noun." This yields entries $\mathcal{B} = \{(r, h, o, t)\}$, where r, h denote REF and HYP words, $o \in \{=, S, D, I\}$ denotes the alignment operation with '=' denoting a match, and t is the PoS tag. For English, the aligned REF, HYP, and Eval sequences enable direct mapping of errors to words, after which a PoS tagger is applied to the relevant word. This results in error-PoS mappings (Table 2) for Table 1.

However, for several non-Latin script languages such as Tamil, the alignments produced by `sclite` break after insertions or deletions, causing REF, HYP, and Eval entries to shift out of correspondence, as illustrated in Table 3[5]. Although the error types (S, I, D) are often identified correctly, the associated words are no longer properly aligned. Similarly, for Russian (Table 4), a deletion causes subsequent words to shift, incorrectly aligning tokens such as Латинская with a deletion and Америка with Латинская, thereby propagating erroneous PoS mappings. For Arabic (Table 5), the deletion of a punctuation symbol disrupts the remaining alignment, leading to incorrect pairings such as سيكذب with لأجلك. Such misalignments make downstream linguistic analysis unreliable, motivating the need for the proposed alignment correction algorithm. Similar alignment failures for Kannada, Greek and Hindi are shown in Tables 8, 9 and 10 (Appendix C). Due to the incorrect alignment produced by `sclite`, it becomes ambiguous which Eval tag corresponds to which REF-HYP word pair, leading to erroneous PoS mappings when a PoS tagger is applied directly. As illustrated in Table 3, (a), Table 4, (a), and Table 5, (a), the HYP sentence becomes left or right-shifted relative to the REF sentence, making the resulting PoS analysis unreliable. Consequently, automated linguistic error analysis becomes infeasible. To address this, we propose a character-spacing aware modified Needleman-Wunsch based alignment mechanismn.

[5]Transliterators used: Indic languages (Aksharamukha) and other languages (Lexilogos).

3.2 Needleman-Wunsch based alignment

Consider two sequences; the reference sequence in REF, $R = (r_1, r_2, \ldots, r_N)$ and the hypothesis sequence in HYP, $H = (h_1, h_2, \ldots, h_M)$ where each r_i corresponds to the ith word in REF and each h_i corresponds to the ith word in HYP. Let c_{sub}, c_{ins}, and c_{del} denote the costs assigned to S, I and D errors respectively. We define a dynamic programming matrix $D \in \mathbb{R}^{(N+1)\times(M+1)}$, where $D(i, j)$ represents the alignment cost between the prefixes r_i and h_j. The alignment is obtained by minimizing a weighted Levenshtein distance using the following recurrence.

$$D(i,j) = \min \begin{cases} D(i-1, j-1) + c_{\text{sub}}(r_i, h_j) \\ D(i-1, j) + c_{\text{del}} \\ D(i, j-1) + c_{\text{ins}} \end{cases} \tag{1}$$

The substitution cost is defined as,

$$c_{\text{sub}}(r_i, h_j) = \begin{cases} 0 & \text{if } r_i = h_j \quad \text{(correct match)} \\ 1 & \text{if } r_i \neq h_j \quad \text{(substitution)} \end{cases}$$

The initialization conditions for the first row and first column are $D(0,0) = 0$, $D(i,0) = i$, $D(0,j) = j$. Following this, the formula for filling the rest of the entries of the matrix is given by (1). Let $\mathcal{A} = \{(r_k, h_k, o_k)\}_{k=1}^{L}$ denote the set of alignments between the reference and

Alignment using sclite	
REF	Латинская Америка — это регион с наибольшим в мире неравенством.
HYP	Латинская Америка — это регион с наибольшим ** мирьем неравенства.
Eval	D S S
Alignment using proposed algorithm	
REF	Латинская Америка — это регион с наибольшим в мире неравенством.
HYP	Латинская Америка — это регион с наибольшим ** мирьем неравенства.
Eval	D S S

ref_word	hyp_word	tag	PoS
Латинская	-	D	adj
Америка	Латинская	S	propn
-	Америка	S	punct

(a) PoS tags after sclite alignment

ref_word	hyp_word	tag	PoS
в	**	D	adp
мире	мирьем	S	noun
неравенством.	неравенства.	S	noun

(b) PoS tags after proposed alignment

Table 4: Comparison between `sclite` and proposed alignment for Russian ASR ("Latinskaja Amerika — èto region s naibol šim v mire neravenstvom.": 'Latin America is the region with the greatest inequality in the world')

Alignment using sclite	
REF	من سيكذب لأجلك ، سيكذب عليك.
HYP	سيكذب عليك** ما سيكذب لأجلك
Eval	S D S
Alignment using proposed algorithm	
REF	من سيكذب لاجلك ، سيكذب عليك.
HYP	ما سيكذب لاجلك ** سيكذب عليك
Eval	S D S

(a)

ref_word	hyp_word	tag	PoS
ما	من	S	prep
سيكذب	-	D	verb
سيكذب	لأجلك	S	noun

(b)

ref_word	hyp_word	tag	PoS
ما	من	S	prep
،	****	D	punc
عليك	عليك.	S	pron

Table 5: Arabic ASR alignment using `sclite` (a) and proposed method (b) ("mana sayakadḧaba laajalaka , sayakadḧaba älayaka.": 'whoever lies for you will lie to you')

hypothesis sequences, where r_k and h_k represent the kth aligned reference and hypothesis tokens respectively, and $o_k \in \{=, S, D, I\}$ denotes match, substitution, deletion, or insertion operation corresponding to the REF and HYP words r_k and h_k respectively. Here L denotes the number of alignment operations. The minimum cost alignment is then obtained by minimizing the Needleman-Wunsch edit distance over the set of alignments consistent with the observed evaluation tag sequence E, i.e.,

$$\hat{\mathcal{A}} = \arg\min_{\mathcal{A} \in \Omega_E} D_{\mathrm{NW}}(\mathcal{A}), \tag{2}$$

where Ω_E denotes the set of admissible alignments satisfying the external S/I/D tag constraints and $D_{\mathrm{NW}}(\mathcal{A})$ is defined as the cumulative edit cost, $D_{\mathrm{NW}}(\mathcal{A}) = \sum_{k=1}^{|\mathcal{A}|} c(o_k)$, with $c(o_k)$ representing the corresponding match, substitution, insertion, or deletion cost corresponding to the kth alignment operation o_k. In order to solve the optimization problem in (2), there are two steps to be followed. The first is the forward step comprising the filling of a cost matrix. Next, a backtracking procedure is applied on the matrix in order to obtain the optimal sequence alignment between REF and HYP. Consider the REF and HYP entries given in Table 6. There is one deleted word in HYP, namely the second word, and one inserted word in HYP in the fourth position. Our objective is to align the HYP to REF with minimal cost of (S, D, I).

REF	he is going ** home
HYP	he ** going to home
Eval	D I

Table 6: An example for illustration of Needleman-Wunsch based alignment

Matrix construction. The cost matrix is constructed using (1) which can be illustrated using the example given in Table 6. First, the REF tokens are filled along the rows and the HYP tokens are filled along the columns as shown in Fig. 1. Proceeding in this manner, each entry in the row is computed using the same recurrence by considering three possibilities: matching/substitution (diagonal), deletion (up), and insertion (left). This ensures that $D(i, j)$ always stores the minimum cost alignment between the corresponding prefixes. For the example in Table 6, the fully constructed matrix is shown in Fig. 1.

Backtracking. The backtracking algorithm used to solve (2) can be explained as follows. Starting from the terminal state (N, M), the alignment path is constructed by tracing the predecessor state that

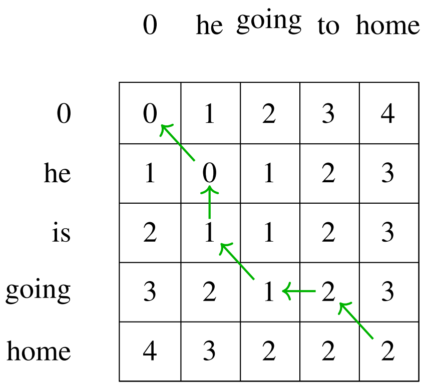

Figure 1: Dynamic programming matrix for global alignment between REF and HYP using our proposed algorithm. The highlighted path shows the backtracking corresponding to the minimum edit distance.

produced the minimum cost in (1). At each step (i, j), one of the following transitions is selected: (1) $(i, j) \rightarrow (i-1, j-1)$ corresponding to a match or substitution, (2) $(i, j) \rightarrow (i-1, j)$ corresponding to a deletion, or, (3) $(i, j) \rightarrow (i, j-1)$ corresponding to an insertion based on $\min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\}$. The process continues until the initial state $(0, 0)$ is reached. In the event of all the neighboring entries having equal cost $(D(i-1, j-1), D(i-1, j), D(i, j-1))$ with each predecessor equally probable, the tie breaking priority is as follows: match('=') > I > D > S. The recovered alignment is represented as $\hat{\mathcal{A}} = \{(r_k, h_k, o_k)\}_{k=1}^{L}$ where r_k and h_k denote the aligned reference and hypothesis tokens respectively, and $o_k \in \{=, S, D, I\}$ denotes the corresponding alignment operation. The detailed illustration of the entire algorithm for the example in Table 6 is given in Appendix B. The optimal alignment is obtained by following the path traced in Fig. 1 leading to the alignment in Table 6.

Summarizing, a diagonal move $(i, j) \rightarrow (i-1, j-1)$ denotes either a correct match ($r_i = h_j$) or a substitution ($r_i \neq h_j$), an upward move $(i, j) \rightarrow (i-1, j)$ denotes a deletion in HYP, and a leftward move $(i, j) \rightarrow (i, j-1)$ denotes an insertion in HYP. Thus, the Needleman–Wunsch algorithm recovers an alignment that identifies deletions and insertions preserving the global structure of the sentence.

Post-processing modification. After alignment using the modified Needleman-Wunsch algorithm, we perform a post-processing step. Specifically, consecutive insertion-deletion pairs (I,D) or deletion-insertion (D,I) pairs are merged into a substitution S, reducing the number of edit operations by one. Such merging avoids artificially inflating the error count by treating a insertion-deletion pair as two separate errors instead of a single substitution error. For example, consider the following REF and HYP sequences: REF: அவன் வீட்டிற்கு சென்றான் ("avan vīttirku senrān": 'he went home'); HYP: அவன் வீடு சென்றான். Suppose the initial Needleman-Wunsch alignment produces $\mathcal{A}$ = (அவன்,அவன்,=), (வீட்டிற்கு,-,D), (-,வீடு,I), (சென்றான்,சென்றான்,=) . Here, the alignment contains two non-match operations: (D,I). They are merged into a single substitution operation S. Hence, the final alignment becomes $\mathcal{A}_{NW}$ = (அவன்,அவன்,=), (வீட்டிற்கு,வீடு,S), (சென்றான்,சென்றான்,=).

Character width based spacing. The misalignment in many non-Latin scripts arises because `sclite` assumes uniform character widths [6], whereas actual rendered widths vary across characters, as seen in Table 3. For instance, Tamil characters such as ர் occupy less space than wider characters like அ or பா. Since `sclite` inserts a fixed number of '*' characters for missing tokens without accounting for these width variations, the REF and HYP sequences become misaligned, leading to incorrect Eval tag mapping. To ensure proper alignment of REF, HYP, and EVAL sequences, especially for ASR outputs of non-Latin scripts, we employ a character width-based spacing mechanism that accounts for the actual display width of each token rather than its raw character count. This is particularly important for Indic and other non-Latin scripts, where display width may differ significantly due to combining characters, vowel modifiers, and other script-specific rendering behavior. Here, *combining characters* are Unicode symbols attached to a base character rather than appearing independently; for example, vowel characters in Tamil and Hindi are often rendered together with consonants as a single unit, which is characteristic of Abugida scripts. Similarly, *vowel modifiers* are marks displayed above, below, before, or after a consonant to alter its pronunciation. Consequently, two tokens with the same character length may still occupy different display widths.

In Hindi, tokens such as कि and की contain com-

[6]Note that the working of the algorithm is not easy to verify as there does not exist a reliable code base.

bining vowel markers that alter display width despite comparable character counts. Similarly, Arabic characters are rendered in connected cursive forms and may change shape depending on position within a word; for example, كتاب and كتــاب can occupy different rendered widths despite representing similar character sequences. In Cyrillic scripts like Russian also, character widths are non-uniform, e.g., мир and шум contain the same number of characters but occupy different widths.

Hence, standard character-count padding may fail to align REF and HYP tokens correctly because the rendered width of complex non-Latin characters differs from their raw character counts. To address this, we compute token widths using a width-aware rendering function. Let (r, h, o) denote an aligned reference token 'r', hypothesis token 'h', and alignment operation 'o'. The corresponding column width is computed as

$$w = \max\big(\mathrm{width}(r), \mathrm{width}(h), \mathrm{width}(o)\big),$$

where $\mathrm{width}(\cdot)$ denotes the rendered token width. Each element is then padded to width $w + \delta$, where δ is a small spacing constant for readability and clearer word boundaries; in our implementation, $\delta = 1$. This width-aware spacing mechanism ensures consistent alignment across both Latin and non-Latin scripts, thereby improving readability and downstream linguistic analysis. Using the proposed character-spacing-aware modified Needleman–Wunsch alignment algorithm, the examples in Tables 3, 4, and 5 now produce correct PoS mappings in their corresponding subfigures (b). Hence, accurate PoS tag mapping is achieved through the proposed alignment method. For example, in Table 5 (b), the deletion is correctly associated with the missing comma "،", while the substitutions من → ما and عليك → عليك. are aligned correctly. Similar alignment corrections for Hindi, Kannada, and Greek are provided in Appendix C and an ablation study is provided in Appendix D. We now proceed with PoS error analysis.

4 Error Analysis

After alignment, we use PoS taggers, described as a function defined by $\mathcal{F} : \mathcal{E} \rightarrow \mathcal{C}$ where $\mathcal{E}$ is the set of (S, D, I) errors and $\mathcal{C}$ is the set of all Universal PoS tags, namely, *adj*, *adp*, *adv*, *aux*, *cconj*, *det*, *intj*, *noun*, *num*, *part*, *pron*, *propn*, *punct*, *sconj*, *verb*, *x* and, *space*, corresponding to adjective, adposition, adverb, auxiliary, coordinating conjunction, determiner, interjection, noun, numeral, particle, pronoun, proper noun, punctuation, subordinating conjunction, verb, other, and whitespace, respectively. More details about the PoS taggers used is in Appendix A. Note that our focus is on emphasizing the importance of PoS-wise error analysis and the utility of the analysis in improving the performance of any model. Hence we focus on the benchmarking models, such as Transformer and Whisper models mentioned in Section 3. The method itself can be applied to any model. In order to analyze error patterns across three different segmented writing systems, we select Tamil, Russian, and Arabic as representatives of the Abugida, Alphabetic, and Abjad systems, respectively. The (S, D, I) error distributions of the Transformer model on the SPRING INX Tamil dataset and the Whisper model on the Common Voice Russian and Arabic datasets are summarized in Table 7 for six major PoS categories (all PoS categories are provided in Tables 12 and 13 of Appendix E). The column "count" denotes the total number of words for the corresponding PoS in each dataset. The color-coded cells represent the ranking of PoS categories in descending order of total error percentage $\frac{(S+D+I)*100}{count}$. The rankings suggest that error patterns may vary across languages.

For Tamil, approximately 20.98% of noun tokens and 24.05% of proper noun tokens are in error. For Russian, approximately 4.56% of noun tokens and 6.88% of proper noun tokens are erroneous. Proper noun errors are likely due to out-of-vocabulary or rare words. Noun errors in Tamil arise from its agglutinative nature, whereas in Russian they stem from its rich inflectional structure. For instance, the Tamil noun குழந்தை ("child") may appear as குழந்தைக்கு ("to/for the child"), while the Russian noun книга ("book") may appear as книги, книгу, or книгой, where suffix variations alter pronunciation and grammatical meaning. For Arabic, approximately 21.03% of noun tokens and 20.45% of proper noun tokens are erroneous. This can be linked to the morphologically rich nature of Arabic, where words undergo root-pattern transformations and affixation. For example, the noun كتاب ("book") may appear as الكتاب ("the book") or كتابه ("his book"), introducing pronunciation and spelling variations. In contrast, verbs, pronouns, adjectives, and adverbs exhibit relatively fewer errors, likely due to their higher frequency and more regular usage patterns, enabling the Transformer to learn more robust rep-

PoS	count	Tamil (Abugida)			count	Russian (Alphabetic)			PoS	count	Arabic (Abjad)		
		S	D	I		S	D	I			S	D	I
noun	18905	15.4	2.4	3.2	23934	4	0.3	0.2	noun	16677	21.5	0.2	0.1
propn	15053	17.5	2.9	3.7	4158	6.1	0.5	0.3	verb	10742	12.7	0.2	0.1
verb	10365	13.7	1.7	2.6	14020	3.6	0.6	0.2	propn	8918	11.7	0.2	0.1
adj	5316	10.6	2.5	3.1	9490	3.1	0.2	0.2	adj	2380	23.9	0.5	0.1
adv	5245	10.7	1.9	3.6	5512	2.4	0.4	0.2	prep	5378	4.9	0.1	0.1
pron	4816	10.6	3.2	4.2	8019	2.9	0.3	0.1	sconj	1680	4.8	0.2	0.1

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6

Table 7: Error distributions of Transformer across Tamil, Russian, and Arabic data, ranked by PoS frequency. Each cell reports the percentage of errors for a PoS category, computed as $\frac{\text{\# errors}}{\text{\# occurrences of the PoS}} \times 100$.

resentations. The PoS error distribution for the languages Hindi, Kannada, Greek and English are discussed in Appendix E. Further, we show how PoS-wise error distributions shift across the models Transformer, LSTM and Conformer for Tamil ASR in Appendix F. In the next section, we propose a PoS-aware Transformer leveraging the PoS-wise error analysis to improve performance.

5 Proposed PoS aware Transformer

At decoding step i, the decoder attends to encoder outputs to identify acoustic frames relevant for predicting target token. Since the target PoS tag is known during training, we use this information to modulate decoder cross-attention based on empirical PoS-wise error distributions, assigning higher importance to frequently error-prone categories. The standard scaled dot-product attention is,

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ denote query, key, and value matrices. We introduce a PoS-dependent weight $w_{\text{POS}(j)}$ for each encoder position j, modifying attention as

$$\tilde{A}_{ij} = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_j^\top}{\sqrt{d_k}} \cdot w_{\text{POS}(j)}\right)$$

Now, the attended output becomes $\mathbf{O}_i = \sum_j \tilde{A}_{ij}\mathbf{V}_j$. Thus, attention scores are reweighted according to syntactic importance inferred from PoS tags. The model is trained using the standard joint CTC-attention objective with label smoothing: $\mathcal{L}_{\text{total}} = \lambda_{\text{CTC}}\mathcal{L}_{\text{CTC}} + (1 - \lambda_{\text{CTC}})\mathcal{L}_{\text{Attn}}$. Here, d_k denotes the key-vector dimensionality and $\lambda_{\text{CTC}} \in [0, 1]$ controls the trade-off.

Experimental setup. We apply PoS-aware training for Tamil, Arabic, and Russian using a Transformer model. The PoS weights are chosen from empirical PoS-wise error distributions by assigning larger weights to categories contributing disproportionately to decoding errors. Further details on the experimental setup and PoS weights are provided in Appendix G. We compare a baseline model without PoS against the proposed PoS-aware Transformer. The architecture is illustrated in Fig. 2 of Appendix H.

Performance improvement. Tamil ASR results show that PoS aware Transformer consistently reduces WER, character error rate (CER), token error rate (TER) and sentence error rate (S.Err), improving WER/ CER/ TER/ S.Err from 23.3/ 8.7/ 16.2/ 73.9 to 22.0/ 8.3/ 15.2/ 67.1. Similarly, for Arabic, WER/ CER/ TER improve from 41.6/ 17.4/ 22.8 to 41.4/ 16.6/ 21.9, while for Russian they improve from 9.7/ 2.9/ 4.1/ 37.9 to 9.2/ 2.7/ 3.8/ 36.0. This indicates that PoS-aware training contributes to more accurate recognition across both languages. More details of PoS-wise error reduction across PoS categories are in Appendix H.

6 Conclusion and Future Scope

In this work, we addressed the problem of incorrect REF-HYP alignment in ASR outputs, particularly for non-Latin scripts, which hinders automated PoS-wise error analysis. We showed that `sclite` fails to align many non-Latin scripts, and proposed a character-spacing aware modified Needleman-Wunsch algorithm that generalizes across both Latin and non-Latin scripts. Using the corrected alignments, we performed automated PoS-wise ASR error analysis and incorporated PoS error statistics into PoS-aware decoding. The promising results indicate that leveraging PoS-wise error analysis for ASR architecture design merits future research.

Limitations

The proposed alignment algorithm is primarily designed for segmented writing systems in which words are explicitly separated by whitespace. Consequently, the method is not directly applicable to non-segmented scripts such as Chinese, Japanese, or Thai, where token boundaries are not explicitly marked. Extending the alignment framework to such languages would require an additional word segmentation stage, whose errors may further influence alignment quality and downstream PoS analysis.

Additionally, the PoS-wise error analysis framework depends heavily on the quality of the underlying PoS tagger. In low-resource languages or languages with limited NLP tools, the PoS tagger may produce inaccurate or overly coarse-grained annotations. For instance, in the Kannada experiments, a large number of words were assigned to the *X* category, indicating that the tagger was unable to confidently determine the appropriate syntactic class. Such misclassifications can affect the reliability of the resulting PoS-wise ASR error statistics and PoS-aware transformer training.

Furthermore, the proposed PoS weighting strategy was an early result to show the utility of PoS-wise ASR error analysis. Hence further improvement is possible.

References

Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber. 2020. Common voice: A massively-multilingual speech corpus. In *Proceedings of the twelfth language resources and evaluation conference*, pages 4218–4222.

Kaushal Bhogale, Abhigyan Raman, Tahir Javed, Sumanth Doddapaneni, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. 2023. Effectiveness of mining audio and text pairs from public data for improving asr systems for low-resource languages. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Chantana Chantrapornchai and Aphisit Tunsakul. 2021. Information extraction tasks based on bert and spacy on tourism domain. *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, 15(1):108–122.

Priyanjana Chowdhury, Nabanika Sarkar, Sanghamitra Nath, and Utpal Sharma. 2024. Analyzing the effects of transcription errors on summary generation of bengali spoken documents. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 23(9):1–28.

Peter T Daniels and William Bright. 1996. *The world's writing systems*. Oxford University Press.

Zar Zar Hlaing, Ye Kyaw Thu, Thepchai Supnithi, and Ponrudee Netisopakul. 2022. Improving neural machine translation with pos-tag features for low-resource language pairs. *Heliyon*, 8(8).

Mohammad Isa Irawan, Imam Mukhlash, Abduh Rizky, and Alfiana Ririsati Dewi. 2019. Application of needleman-wunch algorithm to identify mutation in dna sequences of corona virus. In *Journal of Physics: Conference Series*, volume 1218, page 012031. IOP Publishing.

Antony Alexander James and Parameswari Krishnamurthy. 2025. Pos-aware neural approaches for word alignment in dravidian languages. In *Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL 2025)*, pages 154–159.

Divyanshu Kakwani, Anoop Kunchukuttan, Satya Vamsi Krishna Golla, NC Gokul, Pushpak Bhattacharyya, and Mitesh M Khapra. 2020. Indicnlp-suite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Findings of EMNLP*.

Ngoc-Tien Le, Christophe Servan, Benjamin Lecouteux, and Laurent Besacier. 2016. Better evaluation of asr in speech translation context using word embeddings. In *Interspeech 2016*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Lester James Miranda, Ákos Kádár, Adriane Boyd, Sofie Van Landeghem, Anders Søgaard, and Matthew Honnibal. 2022. Multi hash embeddings in spacy. *arXiv e-prints*, pages arXiv–2212.

Taniya Mishra, Andrej Ljolje, and Mazin Gilbert. 2011. Predicting human perceived accuracy of asr systems. In *INTERSPEECH*, pages 1945–1948.

Andrew Cameron Morris, Viktoria Maier, and Phil D Green. 2004. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *Interspeech*, pages 2765–2768.

National Institute of Standards and Technology (NIST). 2000. Sclite scoring tool. ftp://jaguar.ncsl.nist.gov/current_docs/sctk/doc/sclite.htm. Speech Recognition Scoring Toolkit (SCTK).

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Jan Hajic, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the twelfth language resources and evaluation conference*, pages 4034–4043.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. Camel tools: An open source python toolkit for arabic natural language processing. In *Proceedings of the twelfth language resources and evaluation conference*, pages 7022–7032.

Martha Maria Papadopoulou, Anna Zaretskaya, and Ruslan Mitkov. 2021. Benchmarking asr systems based on post-editing effort and error analysis. In *Proceedings of the Translation and Interpreting Technology Online Conference*, pages 199–207.

Phoebe Parsons, Knut Kvale, Torbjørn Svendsen, and Giampiero Salvi. 2025. Match ‘em: Multi-tiered alignment for error analysis in asr. In *Proceedings of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025)*, pages 440–447.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108.

Nithya R, Malavika S, Jordan F, Arjun Gangwar, Metilda N J, S Umesh, Rithik Sarab, Akhilesh Kumar Dubey, Govind Divakaran, Samudra Vijaya K, and Suryakanth V Gangashetty. 2023. Spring-inx: A multilingual indian language speech corpus by spring lab, iit madras. *Preprint*, arXiv:2310.14654.

Yu Shi, Xi Zhang, and Ning Yu. 2023. Pl-transformer: a pos-aware and layer ensemble transformer for text classification. *Neural Computing and Applications*, 35(2):1971–1982.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Thilo von Neumann, Christoph Boeddeker, Marc Delcroix, and Reinhold Haeb-Umbach. 2025. Word error rate definitions and algorithms for long-form multi-talker speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*.

Lanxiao Wang, Hongliang Li, Heqian Qiu, Qingbo Wu, Fanman Meng, and King Ngi Ngan. 2021. Pos-trends dynamic-aware model for video caption. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(7):4751–4764.

Johannes Wirth and Rene Peinl. 2022. Automatic speech recognition in german: A detailed error analysis. In *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–8. IEEE.

Anuradha Yenkikar, Manish Bali, Rutuja Rajendra Patil, Riddhi Mirajkar, and Tabassum Ara. 2024. Biomedical named entity recognition through spacy: A visual exploration. In *2024 2nd International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, pages 17–22.

A PoS taggers

PoS taggers take as input a sequence of tokens (e.g., a sentence) and predicts a syntactic category label for each token. These predicted labels are treated as the PoS tags of the corresponding words and are subsequently used in our PoS-wise ASR error analysis and PoS-aware decoder training. SpaCy is an open-source library for advanced natural language processing (NLP) in Python and Cython released under the MIT license[7]. It supports 70+ languages, among which we have considered Tamil, Hindi, and Kannada. It can be integrated with many NLP models for PoS tagging and has also been used in several Named Entity Recognition (NER) and Information Extraction (IE) tasks in literature(Yenkikar et al., 2024; Chantrapornchai and Tunsakul, 2021; Miranda et al., 2022).

For Tamil and Hindi, we use Stanza-based PoS taggers integrated through the `spacy-stanza` wrapper[8]. In this setup, SpaCy provides the tokenization interface and document abstractions, while Stanza performs the underlying linguistic annotation. Stanza is a neural NLP toolkit developed by Stanford NLP and is primarily trained on Universal Dependencies (UD) treebanks (Nivre et al., 2020). Its PoS tagger is based on bidirectional Long Short-Term Memory (BiLSTM) networks with character-level representations, enabling robust handling of morphologically rich and agglutinative languages. Stanza additionally performs joint morphological feature prediction together with PoS tagging, improving contextual syntactic understanding.

For Kannada, we use the AI4Bharat IndicBERTv2-based PoS tagging model[9]. The model is built upon transformer-based multilingual

[7] https://github.com/explosion/spaCy/blob/master/LICENSE

[8] `https://github.com/explosion/spacy-stanza`

[9] `https://huggingface.co/ai4bharat/IndicBERTv2-alpha-POS-tagging`

Alignment using sclite	
REF	ಅಹ್ ನಾನ್ಸ ಹುಷಾರ್ ಇದ್ದೇನೆ, ಸೌಖ್ಯ ಆಲ್ಲಿಯೆಲ್ಲ ಮನೆಯಲ್ಲಿ ಎಲ್ಲಾ?
HYP	ಹಾ ನಾನ್ಸ ****************** ಹುಷಾರಿದ್ದೇನೆ ಸೌಖ್ಯ ಅಲ್ಲಿ ಎಲ್ಲ ಮನೆಯಲ್ಲೆಲ್ಲ.
Eval	S D S S S S
Alignment using proposed algorithm	
REF	ಅಹ್ ನಾನ್ಸ ಹುಷಾರ್ ಇದ್ದೇನೆ, ಸೌಖ್ಯ ಆಲ್ಲಿಯೆಲ್ಲ ಮನೆಯಲ್ಲಿ ಎಲ್ಲಾ?
HYP	ಹಾ ನಾನ್ಸ ****** ಹುಷಾರಿದ್ದೇನೆ ಸೌಖ್ಯ ಅಲ್ಲಿ ಎಲ್ಲ ಮನೆಯಲ್ಲೆಲ್ಲ.
Eval	S D S S S S

ref_word	hyp_word	tag	PoS
ಅಹ್	ಹಾ	S	intj
ನಾನ್ಸ	-	D	propn
ಹುಷಾರ್	ನಾನ್ಸ	S	adj
ಇದ್ದೇನೆ,	*********	S	X
ಸೌಖ್ಯ	ಹುಷಾರಿದ್ದೇನೆ	S	noun
ಆಲ್ಲಿಯೆಲ್ಲ	ಸೌಖ್ಯ	S	adv

(a) PoS tags after sclite alignment

ref_word	hyp_word	tag	PoS
ಅಹ್	ಹಾ	S	intj
ಹುಷಾರ್	******	D	adj
ಇದ್ದೇನೆ,	ಹುಷಾರಿದ್ದೇನೆ	S	X
ಆಲ್ಲಿಯೆಲ್ಲ	ಅಲ್ಲಿ	S	adv
ಮನೆಯಲ್ಲಿ	ಎಲ್ಲ	S	noun
ಎಲ್ಲಾ?	ಮನೆಯಲ್ಲೆಲ್ಲ.	S	num

(b) PoS tags after proposed alignment

Table 8: Comparison between `sclite` and proposed alignment for a Kannada ASR example ("ah nānsa huṣār iddēne, saukhya ālliyella maneyalli ellā?": 'Ah, I am fine; is everything well at home there?'); (a), (b): PoS tags after `sclite` and proposed alignment respectively.

Alignment using sclite	
REF	****** Και πρωτομάστορης ήταν ο αδελφός μου.
HYP	και πρωτομά στον Ρησίταν ο αδερφός μου.
Eval	I S S S S
Alignment using proposed algorithm	
REF	****** Και πρωτομάστορης ήταν ο αδελφός μου.
HYP	και πρωτομά στον Ρησίταν ο αδερφός μου.
Eval	I S S S S

ref_word	hyp_word	tag	PoS
-	και	I	cconj
******	πρωτομά	S	punct
Και	στον	S	cconj
πρωτομάστορης	Ρησίταν	S	noun
ήταν	ο	S	verb

(a) PoS tags ater sclite alignment

ref_word	hyp_word	tag	PoS
*****	και	I	cconj
και	πρωτομά	S	cconj
πρωτομάστορης	στον	S	noun
ήταν	Ρησίταν	S	verb
αδελφός	αδερφός	S	noun

(b) PoS tags after proposed alignment

Table 9: Comparison between `sclite` and proposed alignment for a Greek ASR example ("Kai prōtomástorēs ētan o adelphós mou.": 'and my brother was a master craftsman'); (a), (b): PoS tags after `sclite` and proposed alignment respectively.

Alignment using sclite	
REF	डर आ रहा है कि ****** हमलोग जाएंगे फिर कुछ होगा की इतना
HYP	डर आ रहा है कि हम लोग जाएंगे फिर कुछ होगा की इतना
Eval	I S
Alignment using proposed algorithm	
REF	डर आ रहा है कि ****** हमलोग जाएंगे फिर कुछ होगा की इतना
HYP	डर आ रहा है कि हम लोग जाएंगे फिर कुछ होगा की इतना
Eval	I S

ref_word	hyp_word	tag	PoS
-	डर	I	noun
डर	आ	S	noun

(a) PoS tags after sclite alignment

ref_word	hyp_word	tag	PoS
******	हम	I	pron
हमलोग	लोग	S	pron

(b) PoS tags after proposed alignment

Table 10: Comparison between `sclite` and proposed alignment for a Hindi ASR example ("ḍara ā rahā hai ki hamalōga jāēṁgē phira kucha hōgā kī itanā": 'there is fear that if we go, something may happen'); (a), (b): PoS tags after `sclite` and proposed alignment respectively.

contextual representations specifically optimized for Indic languages. It leverages IndicBERTv2

embeddings trained on large-scale multilingual Indic corpora and fine-tuned for PoS tagging tasks.

For Russian, we use the PoS tagger available through spaCy[10], which is trained using UD-style annotations and optimized for efficient syntactic parsing and contextual PoS prediction.

For Arabic, we employ CAMeL Tools[11], a comprehensive Arabic NLP toolkit developed by CAMeL Lab. CAMeL Tools provides modules for tokenization, morphological analysis, disambiguation, and PoS tagging specifically tailored for Arabic, which is a morphologically rich language with extensive inflectional and clitic-based variations. The CAMeL PoS tagger utilizes contextual morphological disambiguation together with pre-trained language representations and linguistically informed analyzers to generate accurate syntactic labels for Modern Standard Arabic (MSA).

For Greek, we use the `gr-nlp-toolkit`[12], which provides linguistic processing modules specifically designed for Modern Greek. The toolkit supports tokenization, lemmatization, dependency parsing, and PoS tagging, and is trained on Greek corpora annotated under the UD framework. Its contextual tagging approach effectively captures the rich inflectional morphology and syntactic structure of Greek.

Most of the PoS taggers used in this work are trained on UD treebanks (Nivre et al., 2020), which provide a consistent multilingual annotation framework based on Universal PoS tags (UPoS). The use of UPoS enables cross-lingual consistency in syntactic analysis and facilitates comparative PoS-wise ASR error evaluation across multiple languages with diverse grammatical structures.

B Proposed Needleman-Wunsch based alignment example

B.1 Matrix construction

Consider the row corresponding to r_2 = "is" in Fig. 1. This row computes $D(2, j)$ for all j by comparing the prefix (he, is) with increasing prefixes of the hypothesis.

For example, consider the computation of $D(2, 2)$, which aligns (he, is) with (he, going). Using (1),

$$D(2,2) = \min \begin{cases} D(1,1) + c_{\text{sub}}(\text{is}, \text{going}) \\ D(1,2) + 1 \\ D(2,1) + 1 \end{cases}$$

Since is $\neq$ going, we have $c_{\text{sub}}(\text{is}, \text{going}) = 1$. Substituting the values from the matrix:

$$D(2,2) = \min \begin{cases} 0+1 \\ 1+1 \\ 1+1 \end{cases} = 1.$$

Similarly, $D(2, 1)$ aligns (he, is) with (he):

$$D(2,1) = \min \begin{cases} D(1,0) + c_{\text{sub}}(\text{is}, \text{he}) \\ D(1,1) + 1 \\ D(2,0) + 1 \end{cases}$$
$$= \min(1+1,\ 0+1,\ 2+1) = 1.$$

which corresponds to deleting "is".

B.2 Backtracking

Consider the example shown in Fig. 1. Starting from $(i, j) = (4, 4)$, the reference and hypothesis tokens are both `home`. Since $D(4, 4) = D(3, 3) + 0$, the current cost could only have been obtained from the diagonal cell through a zero-cost match transition during the matrix filling stage. Hence, we move diagonally to $(3, 3)$, appending $(\text{home}, \text{home}, =)$ to the alignment set $\mathcal{A}$. At $(3, 3)$, the tokens are `going` and `to`, which do not match. We now compare the neighboring cells: $D(2, 3) = 2$, $D(3, 2) = 1$, $D(2, 2) = 1$. Since the leftward transition produces the minimum valid cost corresponding to an insertion operation, $D(3, 3) = D(3, 2) + 1$, we move left to $(3, 2)$, appending $(-, \text{to}, I)$ to $\mathcal{A}$. Next, at $(3, 2)$, the tokens `going` and `going` match. Since $D(3, 2) = D(2, 1) + 0$, the minimum cost must have originated from the diagonal predecessor. Therefore, we move diagonally to $(2, 1)$, appending $(\text{going}, \text{going}, =)$. At $(2, 1)$, the tokens `is` and `he` do not match. Comparing neighboring cells shows that the top transition satisfies $D(2, 1) = D(1, 1) + 1$, corresponding to a deletion. Hence, we move upward to $(1, 1)$, appending $(\text{is}, -, D)$. Finally, at $(1, 1)$, the tokens `he` and `he` match and satisfy $D(1, 1) = D(0, 0) + 0$. Thus, we move diagonally to $(0, 0)$, appending $(\text{he}, \text{he}, =)$. The resulting alignment set is therefore $\mathcal{A} = \{(\text{home}, \text{home}, =), (-, \text{to}, I), (\text{going}, \text{going}, =), (\text{is}, -, D), (\text{he}, \text{he}, =)\}$. Hence the optimal alignment is given by the sequence in Table 6.

[10] https://github.com/explosion/spaCy
[11] https://github.com/CAMeL-Lab/camel_tools
[12] https://github.com/nlpaueb/gr-nlp-toolkit

C Alignment illustration for Hindi, Kannada, and Greek

For all alignments, we set the values of c_{ins} and c_{del} as 1. We use the Transformer model with 12 encoder and 6 decoder blocks[13] for Kannada and Hindi while we use the finetuned Whisper small model for Greek language[14].

The datasets and pretrained models used in this work are publicly available under their respective licenses. The Mozilla Common Voice datasets are distributed under the CC0-1.0 license, while the SPRING-INX datasets are released under the CC BY 4.0 license. All Whisper-based pretrained and fine-tuned models used in our experiments are distributed under the Apache-2.0 license. All assets were used in accordance with their respective terms of use and distribution.

Kannada. As illustrated in Table 8, the `sclite` alignment fails to preserve correct word boundaries after the deletion error, leading to a cascade of incorrect substitutions. In the reference sentence, the word ಹುಷಾರ್ is deleted in the hypothesis; however, `sclite` incorrectly shifts subsequent alignments and maps unrelated words against each other. Consequently, PoS mappings become unreliable, with words such as ನಾನ್ಸ being aligned with a deletion and ಹುಷಾರ್ being incorrectly mapped to ನಾನ್ಸ. This error propagation further causes semantically unrelated substitutions such as ಸೌಖ್ಯ → ಹುಷಾರಿದ್ದೇನೆ. In contrast, the proposed alignment algorithm preserves proper token boundaries after the deletion occurs. The deleted token is correctly associated with ಹುಷಾರ್, while subsequent substitutions such as ಆಲ್ಲಿಯೆಲ್ಲ → ಅಲ್ಲಿ are aligned more accurately. As a result, the PoS-wise mappings become considerably more reliable and interpretable for downstream linguistic error analysis.

Greek. Table 9 demonstrates that the `sclite` alignment produces severe alignment drift after the insertion error at the beginning of the sentence. The inserted word και causes subsequent words to be incorrectly shifted, leading to erroneous substitutions across the remainder of the utterance. As a result, unrelated mappings such as Και → στον and ήταν → ο are produced. This also corrupts the PoS assignments, with several substitutions incorrectly tagged as *punct*. In contrast, the proposed alignment algorithm preserves local word structure despite the insertion. The inserted token is correctly isolated, while meaningful substitutions such as αδελφός → αδερφός are aligned appropriately. Consequently, the resulting PoS-wise mappings are substantially cleaner and better reflect the true ASR errors present in the hypothesis.

Hindi. As shown in Table 10, the `sclite` alignment struggles to handle the insertion of the split token हम लोग corresponding to the reference token हमलोग. Due to this insertion, `sclite` propagates alignment errors toward the beginning of the utterance, incorrectly aligning unrelated neighboring words. For example, words such as डर are falsely marked as substitutions even though they are correctly recognized. This results in misleading PoS assignments such as *noun*. In contrast, the proposed alignment algorithm preserves the correct alignments for the unaffected words and localizes the error around the actual mismatch. The insertion corresponding to हम is correctly identified, while the substitution हमलोग → लोग is aligned properly. Therefore, the proposed method yields substantially more accurate PoS-wise error mappings and avoids the cascading alignment failures observed in the `sclite` output.

D Ablation Study on adaptive character spacing

To evaluate the contribution of adaptive character spacing, we perform an ablation study by removing the spacing correction component and applying only the standard Needleman–Wunsch (NW) alignment algorithm. Although pure NW alignment can still identify approximate substitution regions, the absence of adaptive spacing causes severe alignment drift when insertions or deletions occur between tokens of different lengths.

Table 11 illustrates this issue using a Tamil ASR example. In the pure NW alignment, a large placeholder token (****************) is inserted to represent the initial insertion அதனால,. Since the placeholder length is substantially larger than the actual inserted token, the entire hypothesis sequence becomes right-shifted relative to the reference sequence. This shift propagates through the remaining alignment, causing subsequent words to be matched at incorrect character positions.

As a consequence, downstream PoS mapping becomes corrupted. Several later substitutions collapse entirely into invalid mappings such as `None`.

[13] Transformer 12 layer encoder and 6 layer decoder

[14] https://huggingface.co/mozilla-ai/whisper-small-el

Pure Needleman–Wunsch Alignment	
REF	**************** அவளுக்கொரு பட்டு பாவாடை தைக்கணும்ன்னு சொல்லி, ஒரு யோசனை இருக்கு.
HYP	அதனால, அவளுக்கு ஒரு பட்பாவோட தைக்கணும்னு சொல்லி, ஒரு யோசனை இருக்கு.
Eval	I S S S S
Proposed Alignment with Adaptive Character Spacing	
REF	********* அவளுக்கொரு பட்டு பாவாடை தைக்கணும்ன்னு சொல்லி, ஒரு யோசனை இருக்கு.
HYP	அதனால, அவளுக்கு ஒரு பட்பாவோட தைக்கணும்னு சொல்லி, ஒரு யோசனை இருக்கு.
Eval	I S S S S

Pure Needleman–Wunsch			
ref_word	hyp_word	tag	PoS
***************	அதனால	I	adv
அவளுக்கொரு	அவளுக்கு	S	pron
பட்டு	ஒரு	S	noun
பாவாடை	பட்பாவோட	S	noun
தைக்கணும்ன்னு	None	S	verb

(a) PoS mapping after pure NW alignment

Proposed Alignment			
ref_word	hyp_word	tag	PoS
**************	அதனால,	I	adv
அவளுக்கொரு	அவளுக்கு	S	pron
பட்டு	ஒரு	S	noun
பாவாடை	பட்பாவோட	S	noun
தைக்கணும்ன்னு	தைக்கணும்னு	S	verb

(b) PoS mapping after proposed alignment

Table 11: Comparison between pure Needleman-Wunsch alignment and the proposed character-spacing-aware modified Needleman-Wunsch alignment for a Tamil ASR example. Without adaptive spacing, alignment drift causes several substitutions to collapse into incorrect or missing mappings, leading to shifted PoS assignments and multiple `None`/`UNK` entries. The proposed method preserves token correspondence and enables reliable PoS-aware ASR error analysis.

Thus, even though some substitution locations are approximately identified, the resulting PoS analysis becomes unreliable because the token boundaries are no longer synchronized between the reference and hypothesis sequences.

In contrast, the proposed alignment algorithm dynamically adjusts character spacing to preserve token boundary consistency after insertions and deletions. As shown in Table 11, this prevents cascading right-shift errors and enables semantically meaningful alignments. Words such as பாவாடை and தைக்கணும்ன்னு are correctly aligned with their corresponding hypothesis tokens, resulting in accurate noun- and verb-level PoS assignments. This demonstrates that adaptive character spacing is essential not only for improving visual alignment quality, but also for obtaining reliable downstream PoS-aware ASR error analysis.

E PoS-wise error analysis

In all the tabulated results in this section, the **PoS** column denotes the part-of-speech category assigned to the reference or inserted hypothesis token, while **Count** indicates the total number of occurrences of that PoS category in the dataset. The columns **D**, **S**, and **I** represent the number of deletion, substitution, and insertion errors respectively associated with that PoS category. The **Total** (%) column reports the overall number of errors for that PoS category, computed as D+S+I, along with the corresponding error percentage relative to its total count. Note that, for all datasets, the REF and HYP sentences are normalized by removing punctuation marks.

Tables 12 and 13 provide the detailed PoS-wise error analysis for Tamil, Russian and Arabic datasets of which a sub-table is given in Table 7. It shows clear cross-lingual variation in PoS-wise ASR error distributions. Across Tamil and Arabic, content-bearing categories such as nouns, proper nouns, verbs, and adjectives exhibit substantially higher error rates, dominated primarily by substitution errors. Tamil proper nouns and numerals are particularly error-prone, likely due to vocabulary sparsity and morphological complexity. In contrast, Russian demonstrates comparatively lower and more uniform error rates across PoS categories, reflecting the relative regularity of alphabetic orthography. For Arabic, high substitution rates in nominal and verbal categories indicate the impact of orthographic ambiguity and rich morphology. Overall, these findings highlight the importance of incorporating fine-grained PoS-aware linguistic information into ASR modeling and decoding.

Hindi. For Hindi in Table 14, a similar trend is observed where nouns and pronouns account for the highest error counts, followed by verbs. In particular, nouns contribute the largest number of substitution errors, while pronouns exhibit the highest deletion and insertion errors among all PoS categories. Unlike Tamil, Hindi is less morphologically agglutinative but exhibits rich inflectional variations (e.g., gender, number, and case), which

Tamil						Russian					
POS	Cnt	D	S	I	Tot (%)	POS	Cnt	D	S	I	Tot (%)
noun	18905	451	2911	605	20.98	propn	4158	22	252	12	6.88
propn	15053	433	2630	557	24.05	noun	23934	74	964	53	4.56
verb	10365	182	1423	270	18.09	part	2962	42	67	20	4.36
pron	4816	153	508	201	17.90	cconj	3341	49	60	32	4.22
adj	5316	132	562	163	16.12	num	724	9	18	3	4.14
adv	5245	102	563	191	16.32	verb	14020	36	499	22	3.97
part	3863	87	338	103	13.67	adj	9490	22	294	16	3.50
num	1748	74	366	66	28.95	pron	8019	26	233	9	3.34
aux	2789	37	284	73	14.13	adv	5512	22	135	10	3.03
adp	1432	25	233	33	20.32	det	4110	9	93	6	2.63
det	2131	29	153	82	12.39	adp	8112	72	80	45	2.43
cconj	188	1	3	2	3.19	sconj	1428	3	29	1	2.31
–	–	–	–	–	–	aux	1152	6	14	4	2.08

Table 12: PoS-wise ASR error statistics for Tamil and Russian datasets. D, S, and I denote deletion, substitution, and insertion errors, respectively. Total (%) denotes the percentage of erroneous tokens within each PoS category.

POS	Cnt	D	S	I	Tot (%)
foreign	31	3	42	0	145.16
abbrev	51	0	16	0	31.37
part_det	18	0	5	0	27.78
adj	2380	12	570	1	24.50
noun	16677	39	3581	7	21.75
verb	10742	18	2112	6	19.88
interj	20	0	3	0	15.00
noun_prop	8918	15	1050	6	12.01
adv	336	2	32	1	10.42
pron_interrog	110	0	11	0	10.00
conj	903	6	74	1	8.97
part	201	0	17	0	8.46
part_focus	39	0	3	0	7.69
pron	903	2	57	1	6.64
pron_dem	822	0	53	1	6.57
pron_rel	1177	1	64	1	5.61
noun_quant	111	0	4	2	5.41
part_neg	1229	4	61	0	5.29
conj_sub	1680	4	81	2	5.18
prep	5378	7	266	3	5.13
part_verb	358	0	18	0	5.03
verb_pseudo	134	0	6	0	4.48
part_voc	479	1	20	0	4.38
adv_rel	79	0	3	0	3.80
adv_interrog	117	0	3	0	2.56
part_interrog	355	0	7	0	1.97

Table 13: PoS-wise ASR error statistics for the Arabic dataset. D, S, and I denote deletion, substitution, and insertion errors, respectively. Total (%) denotes the percentage of erroneous tokens within each PoS category.

POS	Count	D	S	I	Total (%)
noun	16557	320	2461	518	3299 (19.93)
pron	12647	396	1493	586	2475 (19.57)
verb	11885	306	1464	412	2182 (18.36)
aux	10538	279	1162	407	1848 (17.54)
propn	4578	206	1289	276	1771 (38.69)
adp	10502	184	890	340	1414 (13.46)
adj	4635	104	736	144	984 (21.23)
part	5132	195	424	226	845 (16.47)
det	3693	89	401	149	639 (17.30)
sconj	2788	82	310	125	517 (18.54)
adv	1993	40	218	100	358 (17.96)
num	1754	57	136	107	300 (17.10)
cconj	2113	67	148	79	294 (13.91)
intj	409	21	137	17	175 (42.79)
x	91	4	19	0	23 (25.27)

Table 14: PoS-wise error statistics for Hindi.

can still lead to recognition ambiguities. Pronouns, despite being frequent, show a relatively high error rate, possibly due to their phonetic similarity (e.g., “वह”, “यह”) and contextual dependence. Verbs also contribute significantly to errors due to variations in tense, aspect, and agreement markers. Overall, while the Transformer model captures frequent functional categories reasonably well, lexical categories such as nouns remain the primary source of errors in both languages.

Kannada. The error distribution on the Kannada dataset in Table 15 reveals a strong dominance of substitution errors (S) across almost all parts of speech similar to Tamil, Hindi. This is particularly evident in high-frequency categories such as adj, sym, propn, and verb, where substitutions account for the majority of total errors, suggesting confusion among acoustically or morphologically similar words. Notably, pron exhibits high total error rate (100%), driven largely by substitutions, which reflects the short, highly context-dependent nature of pronouns in Kannada. Punct, noun, and adp also show high overall error rates (>60%), pointing to challenges in modeling syntactic boundaries and function-word behavior. In contrast, relatively lower error rates for adv, det, and part suggest that these categories are either less frequent, more acoustically distinct, or more predictable given context. Deletion errors (D) are generally smaller compared to substitutions but become more noticeable in verb and pron, indicating occasional failure to capture entire tokens, possibly

POS	Count	D	S	I	Total (%)
x	6392	402	2997	351	3750 (58.67)
adj	6641	408	2505	399	3312 (49.87)
sym	5563	358	1961	421	2740 (49.25)
propn	4193	318	1761	341	2420 (57.72)
verb	3226	411	1262	275	1948 (60.38)
pron	1029	150	774	107	1031 (100.19)
intj	1702	98	653	111	862 (50.65)
num	1332	132	487	127	746 (56.01)
adp	741	39	401	16	456 (61.54)
part	640	38	233	43	314 (49.06)
aux	540	32	219	45	296 (54.81)
noun	232	15	134	8	157 (67.67)
det	335	16	107	24	147 (43.88)
adv	122	9	31	11	51 (41.80)

Table 15: PoS-wise error statistics for Kannada.

due to coarticulation or rapid speech. Insertions (I) remain comparatively low across categories. Overall, the pattern suggests that the primary limitation of the Transformer model on this dataset lies in fine-grained lexical discrimination under phonetic and morphological variability, which is characteristic of agglutinative languages like Kannada. The high number of 'x' errors indicate that the AI4Bharat PoS tagger has failed to classify the PoS of many of the words in the dataset.

POS	Count	D	S	I	Total (%)
num	65	1	8	0	9 (13.85)
part	210	2	23	3	28 (13.33)
cconj	681	4	67	8	79 (11.60)
pron	1347	10	121	6	137 (10.17)
det	1565	23	123	9	155 (9.90)
sconj	151	2	10	1	13 (8.61)
adp	503	5	29	3	37 (7.36)
aux	700	8	23	5	36 (5.14)
adv	845	5	22	0	27 (3.20)
verb	2240	1	12	2	15 (0.67)
noun	2082	3	10	1	14 (0.67)
propn	180	1	0	0	1 (0.56)
adj	598	0	1	0	1 (0.17)

Table 16: PoS-wise error statistics for Greek.

Greek. Table 16 demonstrates comparatively lower error percentages across all PoS categories. Although substitution errors remain the dominant error type, their magnitudes are much smaller than those observed for Hindi and Kannada. Categories such as *num*, *part*, *cconj*, and *pron* exhibit the highest total error percentages, namely 13.85%, 13.33%, 11.60%, and 10.17%, respectively. In contrast, major lexical categories such as *verb* and *noun* show very low total error percentages of only 0.67%. Similarly, *adj*, *propn*, and *punct* exhibit near-zero error rates. These results suggest that the Greek ASR system achieves relatively stable recognition performance across most grammatical categories, with errors concentrated mainly in smaller functional categories and low-frequency tokens.

POS	Count	D	S	I	Total (%)
noun	8709	50	610	22	682 (7.83)
verb	5507	15	230	16	261 (4.74)
propn	8171	11	144	63	218 (2.67)
pron	5587	18	107	38	163 (2.92)
det	5561	23	77	56	156 (2.81)
adj	2567	12	116	11	139 (5.41)
adp	6134	8	83	31	122 (1.99)
aux	3364	12	58	8	78 (2.32)
adv	1928	6	51	8	65 (3.37)
cconj	2356	8	57	13	78 (3.31)
sconj	1139	2	16	5	23 (2.02)
part	1145	4	10	3	17 (1.48)
num	440	1	9	1	11 (2.50)
intj	212	0	7	3	10 (4.72)
x	15	3	6	0	9 (60.00)

Table 17: PoS-wise error statistics for LibriSpeech test-clean.

LibriSpeech test-clean. Table 17 presents the PoS-wise ASR error statistics for the LibriSpeech test-clean subset. Among all PoS categories, *noun* exhibits the highest total number of errors with 682 errors (7.83%), primarily dominated by substitution errors (610). Although the absolute number of errors is large for nouns, the relative error percentage remains moderate due to the high occurrence count of nouns in the corpus. Verbs and adjectives also show comparatively higher error rates, with total error percentages of 4.74% and 5.41%, respectively. Proper nouns (*propn*) have a lower relative error rate of 2.67%, indicating that the model performs comparatively well on named entities in clean acoustic conditions. Functional categories such as *adp*, *aux*, *part*, and *sconj* exhibit relatively low error percentages, suggesting that frequent syntactic tokens are recognized more reliably. The *x* category shows the highest relative error percentage (60.00%); however, this category contains only 15 tokens and therefore does not significantly affect overall WER. Overall, substitution errors dominate across nearly all PoS categories, indicating that acoustic confusions and lexical replacements are more common than deletions or insertions in clean speech conditions.

LibriSpeech test-other. Table 18 presents the PoS-wise ASR error statistics for the LibriSpeech test-other subset, which is acoustically more challenging than test-clean. Compared to test-clean, almost all PoS categories show noticeably higher error percentages. Nouns again contribute the largest number of total errors with 1342 errors (16.94%), dominated by substitution errors (1206). Verbs

POS	Count	D	S	I	Total (%)
noun	7920	71	1206	65	1342 (16.94)
verb	6275	37	549	50	636 (10.14)
propn	8211	18	384	173	575 (7.00)
pron	6209	56	281	76	413 (6.65)
det	5071	60	219	70	349 (6.88)
adj	2169	21	286	24	331 (15.26)
adp	5555	36	212	66	314 (5.65)
aux	3511	21	161	33	215 (6.12)
adv	2026	9	143	14	166 (8.19)
cconj	2298	19	89	35	143 (6.22)
sconj	1281	4	51	19	74 (5.78)
part	1283	10	34	10	54 (4.21)
num	512	3	17	6	26 (5.08)
intj	291	1	11	3	15 (5.15)
x	20	0	12	0	12 (60.00)

Table 18: PoS-wise error statistics for LibriSpeech test-other.

also exhibit a substantial increase in errors, reaching 636 total errors (10.14%). Adjectives show a relatively high error percentage of 15.26%, indicating increased difficulty in recognizing descriptive lexical items under challenging acoustic conditions. Proper nouns (*propn*) have a total error percentage of 7.00%, which is considerably higher than in test-clean, suggesting greater sensitivity of named entities to acoustic degradation. Function words such as *det*, *adp*, *aux*, and *cconj* also experience increased substitution and insertion errors compared to test-clean. Similar to the clean subset, substitution errors remain the dominant error type across nearly all PoS categories.

Overall, the results indicate that acoustically difficult speech conditions disproportionately affect content-bearing PoS categories such as nouns, verbs, and adjectives.

F Error analysis of different models for Tamil dataset

For comparing PoS error distribution across models for Tamil ASR, we consider the Indic-Conformer model[15] comprising of 17 Conformer blocks of model dimension 512. We further use an LSTM model of standard configuration[16]. A comparison across Transformer, LSTM, and Conformer models in Table 19 reveals consistent patterns in error distribution, while also highlighting architecture-specific weaknesses. Across all three models, nouns and proper nouns remain the most error-prone categories, indicating that lexical variability and agglutinative morphology are primary sources of difficulty in Tamil ASR. Both Transformer and LSTM exhibit similar behavior, with dominating substitution errors and relatively moderate deletion and insertion errors. This suggests that Transformer and LSTM are reasonably effective at capturing token boundaries but struggle with lexical precision. In contrast, Conformer displays a different error profile, characterized by an unusually high deletion errors, especially for nouns, proper nouns, and verbs. This indicates a tendency to omit tokens entirely rather than substitute them, possibly due to alignment issues in longer sequences. Interestingly, significant reduction in insertion errors (often near zero) by Conformer, comes at the cost of aggressive deletions. Pronouns, adjectives, and adverbs show relatively lower error counts across all models, likely due to their higher frequency and more constrained usage patterns, though Conformer exhibits high deletions even in these categories. Overall, while Transformer and LSTM maintain a balanced error distribution dominated by substitutions, Conformer introduces a skewed trade-off between deletions and insertions, suggesting that improvements in acoustic modeling do not necessarily translate to better lexical retention in morphologically rich languages like Tamil.

G PoS weights and model details

For incorporating PoS weights, we use a standard Transformer architecture with a 12-layer encoder and 6-layer decoder[17]. The encoder has a model dimension of 512 with 8 attention heads, with the same configuration for the decoder. Training is performed using the Adam optimizer with a learning rate of 0.002 and a warm-up scheduler with 25k warm-up steps. We use gradient accumulation of 4 and a batch size of 16M bins. SpecAugment is applied with time warping, frequency masking, and time masking to improve robustness. For all experiments, λ_{CTC} is set as 0.3. The weights chosen for SPRING INX-Tamil, Commonvoice Arabic and Russian datasets is given in Table 20. For PoS-aware training, these weights are computed token-wise for each test dataset in prior and stored as a JSON file, which is then used by the transformer. All experiments were conducted on a A100 GPU with 80 GB RAM. The training time is $\sim$ 40 hours, $\sim$ 17 hours, and $\sim$ 58 hours for 100 epochs for Tamil, Arabic, and Russian dataset respectively.

[15] IndicConformer Tamil ASR model

[16] LSTM Tamil ASR model

[17] Transformer 12 layer encoder and 6 layer decoder

POS	count	Transformer			LSTM			Conformer		
		S	D	I	S	D	I	S	D	I
noun	18905	15.4	2.38	3.2	16.61	2.68	2.85	17.7	61.15	0.16
propn	15053	17.47	2.87	3.7	17.39	3.2	3.34	16.73	63.52	4.3
verb	10365	13.73	1.76	2.6	13.62	2.0	2.25	29.76	46.99	0.0001
pron	4816	10.55	3.18	4.17	10.20	3.11	3.55	14.04	72.2	0
adj	5316	10.57	2.48	3.07	11.5	2.56	2.78	13.76	71.01	0
adv	5245	10.73	1.94	3.64	10.73	1.94	2.42	15.46	67.5	0.05

Rank 1 Rank 2 Rank 3 Rank 4 Rank 5 Rank 6

Table 19: Error distributions of different models for SPRING-INX Tamil dataset.

Tamil		Arabic		Russian	
PoS	**Weight**	**PoS**	**Weight**	**PoS**	**Weight**
NOUN	4.0	NOUN	8.0	NOUN	6.0
PROPN	4.0	NOUN_PROP	2.0	VERB	6.0
VERB	2.0	VERB	8.0	ADJ	1.0
AUX	1.0	VERB_PSEUDO	5.0	PROPN	1.0
ADJ	1.5	ADJ	2.5	ADP	2.0
ADV	1.5	ADV	1.5	PRON	2.0
PRON	2.0	PRON	1.5	CCONJ	1.0
ADP	1.0	PRON_DEM	1.5	SCONJ	1.0
DET	1.0	PRON_EXCLAM	1.5	ADV	1.5
CONJ	1.0	PRON_INTERROG	1.5	AUX	1.0
CCONJ	1.0	PREP	2.0	PART	1.0
NUM	1.0	CONJ	1.5	NUM	1.0
PART	1.0	PART	1.0	DET	1.0
SYM	1.0	PART_NEG	1.0	INTJ	1.0
INTJ	1.0	PART_FOCUS	1.0	X	1.0
X	1.0	PART_INTERROG	1.0	SYM	1.0
		PART_FUT	1.0	UNKNOWN	1.0
		NUM	1.0		
		ABBREV	1.0		
		FOREIGN	1.0		
		INTERJ	1.0		

Table 20: PoS-specific weights used for decoder-side PoS-aware attention for Tamil, Arabic, and Russian.

Dataset	Train	Validation	Test
Tamil	56,112	16,699	4,884
Arabic	28,865	10,229	10,508
Russian	26,920	10,282	10,283

Table 21: Number of utterances in the train, validation, and test splits for the evaluated datasets.

The train, validation and test splits for the datasets used in training are provided in Table 21.

H PoS-wise distribution after Decoder PoS fusion

An illustration of the Decoder PoS fusion in the Transformer encoder-decoder architecture is provided in Fig. 2.

Tamil.

Table 22 presents the PoS-wise error analysis for the SPRING-INX Tamil dataset. Categories with a reduction in the number of errors are highlighted in bold. Across most PoS categories, incorporating PoS information in the decoder reduces substitution errors, which remain the dominant error type. Significant improvements are observed for content-bearing categories such as *noun*, *propn*, *verb*, *adv*, and *punct*. For instance, substitution errors for *propn* decrease from 2630 to 2419, corresponding to a relative reduction of approximately 8.02%. Similarly, substitution errors for *verb* reduce from 1423 to 1351 (5.06% reduction), while *noun* substitutions decrease from 2911 to 2807 (3.57% reduction). Categories such as *adv*, *aux*, *num*, and *punct* also exhibit consistent reductions in both insertion and substitution errors. Although a few categories such as *part* show a slight increase in deletions, the overall total error percentage decreases for most major PoS groups. These results suggest that decoder-level PoS conditioning helps the model better capture syntactic and lexical structure,

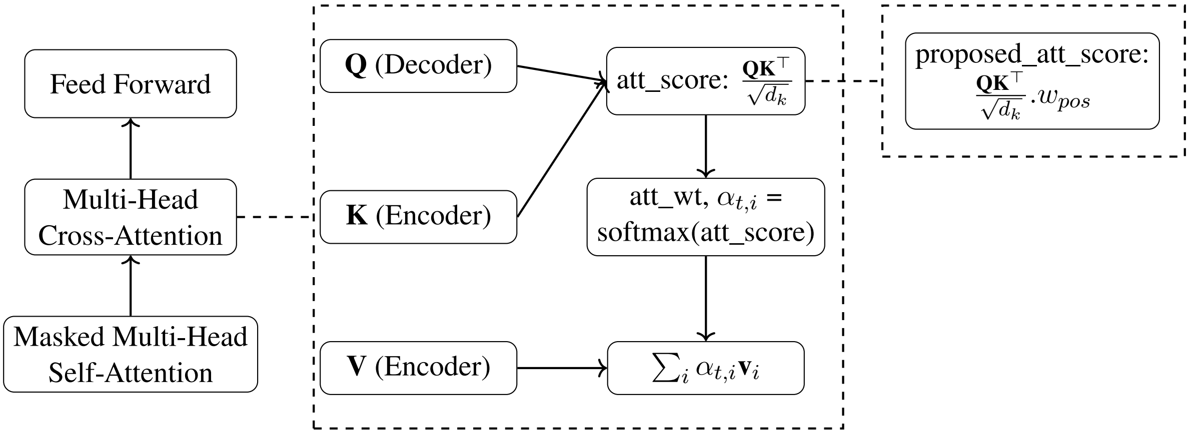

Figure 2: Col-1: Default decoder layers in Transformer; Col-2: cross-attention with attention score computation; Col-3: modified attention score using scalar PoS weights, w_{pos}

PoS	**Count**	**Without PoS**				**Decoder with PoS**			
		D	S	I	Total (%)	D	S	I	Total (%)
noun	18905	451	2911	605	3967 (20.98)	470	**2807**	**546**	3823 (20.22)
propn	15053	433	2630	557	3620 (24.05)	474	**2419**	**532**	3425 (22.75)
verb	10365	182	1423	270	1875 (18.09)	205	**1351**	**240**	1796 (17.33)
pron	4816	153	508	201	862 (17.90)	165	**494**	**161**	820 (17.03)
adj	5316	132	562	163	857 (16.12)	**127**	562	**143**	832 (15.65)
adv	5245	102	563	191	856 (16.32)	117	**524**	**151**	792 (15.10)
part	3863	87	338	103	528 (13.67)	98	338	104	540 (13.98)
num	1748	74	366	66	506 (28.95)	**70**	**351**	**63**	484 (27.69)
aux	2789	37	284	73	394 (14.13)	37	**257**	**64**	358 (12.84)
adp	1432	25	233	33	291 (20.32)	34	**217**	**32**	283 (19.76)
det	2131	29	153	82	264 (12.39)	29	164	**58**	251 (11.78)
cconj	188	1	3	2	6 (3.19)	1	**2**	**1**	4 (2.13)

Table 22: PoS-wise error comparison for SPRING-INX Tamil dataset; without PoS refers to baseline; Decoder with PoS: PoS incorporated in Transformer decoder

thereby improving recognition accuracy and contributing to the overall WER reduction from 23.3 to 22.0. For a graphical comparison of reduction in PoS-wise errors, refer to Fig. 3.

Arabic. The non-UPoS tags used by the Arabic PoS tagger correspond to fine-grained grammatical categories specific to Arabic morphology, described in Table 23. Table 24 presents the PoS-wise error analysis for the Arabic dataset, where lower error counts between the two systems are highlighted in bold. Similar to the Tamil experiments, substitution errors dominate across most PoS categories. Incorporating PoS information into the decoder reduces errors for several major lexical categories such as *noun*, *verb*, *noun_prop*, and *prep*, which contribute significantly to the overall WER. In particular, total errors for *noun* decrease from 6660 to 6519 (2.12% reduction), *noun_prop* decreases from 2781 to 2740 (1.47% reduction), and *prep* decreases from 1980 to 1962 (0.91% reduction). Function-word categories such as *adv*, *part_focus*, and *foreign* also show improvements, mainly through reduced substitution errors, indicating that PoS-aware decoding helps the model better capture syntactic and lexical context during prediction. Although a few categories such as *adj*, *conj_sub*, and *part_interrog* exhibit slight increases in errors, the dominant content-bearing categories show overall improvement, contributing to the WER reduction from 41.6 to 41.4.

Russian. Table 25 presents the PoS-wise error analysis for the Russian dataset. Similar to the Tamil and Arabic experiments, substitution errors remain the dominant error type across most PoS categories. Incorporating PoS information into the decoder reduces total errors for several major lexical and functional categories, particularly *propn*, *noun*, *adp*, *cconj*, *pron*, and *part*.

The largest improvement is observed for *propn*, where total errors decrease from 3754 to 3607, corresponding to a relative reduction of approximately 3.92%. Likewise, *noun* errors reduce from 710 to 673 (5.21% reduction), while *adp* errors decrease from 557 to 535 (3.95% reduction). Categories

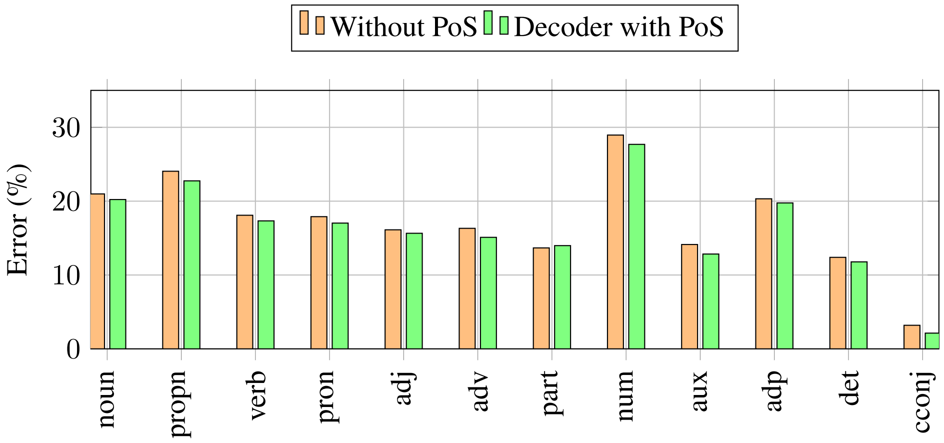

Figure 3: PoS-wise error comparison for SPRING-INX Tamil dataset;x-axis: PoS tags; y-axis: corresponding error.

Tag	Description
noun_prop	Proper noun (names of people, places, organizations)
noun_quant	Quantifier noun indicating quantity or amount
verb_pseudo	Pseudo-verbs or quasi-verbal particles in Arabic grammar
part_det	Determiner particle
part_neg	Negation particle
part_voc	Vocative particle used for addressing
part_verb	Verbal particle associated with tense/aspect constructions
part_focus	Focus or emphasis particle
part_interrog	Interrogative particle used in questions
pron_dem	Demonstrative pronoun
pron_rel	Relative pronoun
pron_interrog	Interrogative pronoun
conj_sub	Subordinating conjunction
adv_rel	Relative adverb
adv_interrog	Interrogative adverb
interj	Interjection
foreign	Foreign or non-Arabic word
abbrev	Abbreviation

Table 23: Fine-grained non-UPoS tags used by the Arabic PoS tagger.

such as *pron* and *part* also exhibit consistent reductions in deletion and substitution errors, leading to overall total error reductions of approximately 12.22% and 8.16%, respectively. The *det* category additionally benefits from reduced substitution and insertion errors, decreasing total errors from 30 to 27.

However, the gains are not uniform across all categories. Certain low-frequency PoS groups such as *adj*, *verb*, and *sconj* show slight increases in total errors, mainly due to increased substitutions or insertions. Nevertheless, across the major content-bearing and syntactically important categories, PoS-aware decoder training generally reduces substitution and deletion errors, contributing to the overall WER improvement from 9.7 to 9.2.

I Statement on Generative AI Use

Generative AI tools were used solely for grammar checking and language refinement.

PoS	**Count**	**Baseline**				**Decoder with PoS**			
		D	S	I	Total (%)	D	S	I	Total (%)
noun	16676	**274**	6197	189	6660 (39.94)	287	**6056**	**176**	**6519 (39.09)**
verb	10741	**244**	4690	175	5109 (47.57)	264	**4679**	**164**	**5107 (47.55)**
noun_prop	8919	**132**	2546	103	2781 (31.18)	147	**2491**	**102**	**2740 (30.72)**
prep	5378	**105**	1817	**58**	1980 (36.82)	126	**1769**	67	**1962 (36.48)**
adj	2380	**28**	**747**	**17**	**792 (33.28)**	31	757	20	808 (33.95)
verb_pseudo	134	13	163	**7**	183 (136.57)	**9**	**159**	8	**176 (131.34)**
part_neg	1229	**39**	586	41	666 (54.19)	55	**571**	**31**	**657 (53.46)**
pron_rel	1177	49	**534**	57	**640 (54.38)**	**44**	543	**56**	643 (54.63)
conj_sub	1680	**21**	**466**	20	**507 (30.18)**	27	468	20	515 (30.65)
pron	903	32	400	**21**	453 (50.17)	33	**388**	26	**447 (49.50)**
conj	903	**36**	370	34	**440 (48.73)**	47	**366**	**30**	443 (49.06)
part_voc	479	**49**	298	32	379 (79.12)	68	**267**	**31**	**366 (76.41)**
part_interrog	355	38	**154**	**52**	**244 (68.73)**	**34**	162	63	259 (72.96)
part_verb	358	**14**	**187**	13	**214 (59.78)**	21	192	**11**	224 (62.57)
abbrev	51	**19**	**164**	**12**	**195 (382.35)**	24	169	13	206 (403.92)
adv	336	19	123	7	149 (44.35)	**18**	**113**	**3**	**134 (39.88)**
pron_interrog	110	14	**64**	**15**	**93 (84.55)**	**13**	67	20	100 (90.91)
part	201	2	**66**	1	**69 (34.33)**	**1**	78	**0**	79 (39.30)
adv_rel	79	9	**41**	**11**	**61 (77.22)**	**8**	55	21	84 (106.33)
adv_interrog	117	9	39	**7**	**55 (47.01)**	**7**	39	10	56 (47.86)
part_focus	39	**2**	33	3	38 (97.44)	5	**29**	**1**	**35 (89.74)**
foreign	31	2	20	0	22 (70.97)	2	**19**	0	**21 (67.74)**
interj	20	2	**11**	5	18 (90.00)	**1**	13	**2**	**16 (80.00)**
part_fut	17	**1**	**8**	1	**10 (58.82)**	2	9	**0**	11 (64.71)
noun_quant	111	0	**7**	**0**	**7 (6.31)**	0	8	1	9 (8.11)
part_det	18	**0**	**1**	**0**	**1 (5.56)**	–	–	–	–

Table 24: PoS-wise error comparison for the Arabic dataset. Baseline refers to the standard Transformer ASR model, while Decoder with PoS denotes Transformer decoder training augmented with PoS information. Boldface indicates lower error counts between the two systems for each metric.

PoS	**Count**	**Baseline**				**Decoder with PoS**			
		D	S	I	Total (%)	D	S	I	Total (%)
propn	60150	416	3007	**331**	3754 (6.24)	**361**	**2886**	360	**3607 (6.00)**
noun	11946	**84**	557	**69**	710 (5.94)	89	**511**	73	**673 (5.63)**
adp	6063	239	**206**	**112**	557 (9.19)	**206**	209	120	**535 (8.82)**
cconj	3141	150	**120**	71	341 (10.86)	**142**	126	**55**	**323 (10.28)**
x	617	9	**221**	**5**	**235 (38.09)**	**5**	232	6	243 (39.38)
part	1430	31	**44**	**23**	98 (6.85)	**16**	45	29	**90 (6.29)**
pron	2015	29	48	**13**	90 (4.47)	**24**	**40**	15	**79 (3.92)**
det	595	**6**	22	2	30 (5.04)	9	**18**	**0**	**27 (4.54)**
verb	572	6	16	**3**	**25 (4.37)**	**4**	16	6	26 (4.55)
adj	90	1	**3**	**0**	**4 (4.44)**	**0**	8	**0**	8 (8.89)
adv	240	2	5	1	8 (3.33)	**1**	**4**	1	**6 (2.50)**
sconj	72	**0**	**0**	1	**1 (1.39)**	1	**0**	1	2 (2.78)
num	2	0	1	0	1 (50.00)	0	1	0	1 (50.00)

Table 25: PoS-wise error comparison for the Russian dataset. Baseline refers to the standard Transformer ASR model, while Decoder with PoS denotes Transformer decoder training augmented with PoS information. Boldface indicates lower error counts between the two systems for each metric.