The Gemma Sutras: Fine-Tuning Gemma 3 for Sanskrit Sandhi Splitting

Samarth P
Computer Science and Engineering
PES University
Bengaluru, India
pes2ug22cs495@pesu.pes.edu

Sanjay Balaji Mahalingam
Computer Science and Engineering
PES University
Bengaluru, India
pes2ug22cs501@pesu.pes.edu

Abstract

Sandhi, the phonological merging of morphemes, is a central feature of Sanskrit grammar. While Sandhi formation is well-defined by Pāṇini's Aṣṭādhyāyī, the reverse task, Sandhi splitting, is substantially more complex due to inherent ambiguity and context-sensitive transformations. Accurate splitting is a critical precursor to tokenization in Sanskrit, which lacks explicit word boundaries and presents densely fused compounds. In this work, we present a data-driven approach, fine-tuning the Gemma-3 4B large language model on a dataset of over 49,000 training and 2,000 test examples of compound words and their morpheme-level decompositions. Leveraging the Unsloth framework with low-rank adaptation (LoRA) and 4-bit quantization, we train the model to predict these splits. Our work yields a scalable, Sandhi-aware system designed to enhance modern NLP pipelines for classical Sanskrit, demonstrating an effective application of LLMs to this linguistic challenge.

1 Introduction

Sanskrit, an ancient language with a vast literary corpus (Kulkarni, 2010; Huet, 2003) and a grammar codified by Pāṇini (Cardona, 1997; Kiparsky, 2009) that is a cornerstone of linguistics (Briggs, 1985), features a key morphological process called Sandhi (संधि). This rule-governed merging of adjacent morphemes (Dave et al., 2021; Rama and Lakshmanan, 2009), illustrated in Figure 1, creates long, uninterrupted compound words. While Sandhi formation is deterministic, the reverse process of splitting, or viccheda (विच्छेद), is significantly more complex due to inherent ambiguity (Aralikatte et al., 2018; Gantayat et al., 2018). This complexity makes effective tokenization, a foundational NLP step, extremely challenging. Naïve tokenizers fail on compounds like तदुपासनीयम् (which must be split to तत् + उपासनीयम्) (Reddy et al., 2018; Bhatt et al., 2024), and even modern subword algorithms like BPE (Sennrich et al., 2016) or WordPiece (Wu et al., 2016; Schuster and Nakajima, 2012) struggle because the transformations disrupt statistical regularities (Li and Girrbach, 2022; Li, 2023). To address this, we frame Sandhi splitting as a data-driven, linguistically-informed pre-tokenization task. We fine-tune a large language model on an annotated dataset to accurately segment these compounds, with our overall approach depicted in Figure 2.

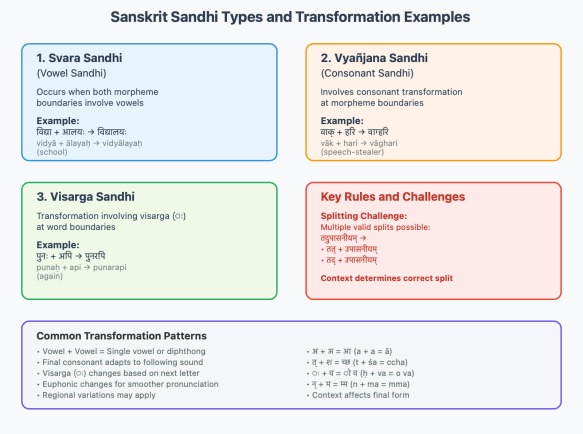

Figure 1: Overview of Sanskrit Sandhi types, common transformation patterns, and key splitting challenges. Refer to Section 1 for discussion.

2 Related Works

Automated Sanskrit Sandhi splitting has progressed through several computational paradigms, as surveyed by Gaikwad and Jatinderkumar (2021) and more recently for deep learning techniques by S et al.. Early approaches were rule-based, grounded in Pāṇini's grammar (Rama and Lakshmanan, 2009; Raja et al., 2014) and exemplified by tools like the JNU Splitter (Mittal, 2010) and INRIA Reader (Huet, 2005; Goyal and Huet, 2013). These systems, however, often exhibit low perfor-

Proceedings of the 9th Widening NLP Workshop, pages 235–241

mance on benchmarks like the SandhiKosh corpus (Bhardwaj et al., 2018) due to the inherent ambiguity of Sandhi. A conceptual shift came with deep learning, which framed the task as a sequence-to-sequence problem Aralikatte et al. (2018).Models like the Double Decoder RNN (DD-RNN) learned transformations directly from character data using a two-stage process (locate split, then reconstruct), a paradigm also explored by Gantayat et al. (2018). This two-stage neural approach was later refined by Dave et al. (2021), whose model first identified a localized "Sandhi window" before decoding, improving efficiency on a large dataset from the UoH corpus (Krishna et al., 2020). Building on these foundations, this work shifts to modern Large Language Models (LLMs). While their application to this specific task is underexplored, we propose that fine-tuning an LLM offers a more generalizable and simpler approach than specialized architectures. We leverage instruction tuning (Ouyang et al., 2022; Wei et al., 2021; Sanh et al., 2021) and parameter-efficient methods (Lialin et al., 2023; Ding et al., 2023) to adapt a model for this nuanced linguistic challenge.

3 Methodology

To address Sandhi splitting, we adopt a supervised fine-tuning approach using the Gemma-3 4B Instruction-Tuned large language model (Gemma Team et al., 2024). The goal is to adapt the model's generative capabilities to split compound Sanskrit words into their morphemic components. Our overall pipeline is summarized in Figure 2.

We selected the Gemma-3 4B variant as its instruction-tuned nature aligns well with our prompt-response task format (Ouyang et al., 2022; Wei et al., 2021), and its 4-billion parameter size offers a practical balance between performance and resource efficiency. The model's Transformer architecture (Vaswani et al., 2017) incorporates features like Rotary Position Embeddings (RoPE) (Su et al., 2024), and its SentencePiece tokenizer (Kudo and Richardson, 2018) supports the Devanagari script.

3.1 Training Objective

The model is trained to generate correct Sandhi splits by framing the task as an instruction-following problem. Each training instance consists of a dialogue where the model must produce a structured output:

> **System**: "Please split the Sandhis"
> **User**: Compound word (e.g., श्रीमद्भगवद्गीता, śrīmadbhagavadgītā) **Assistant**: Correct split (e.g., श्रीमत्+भगवत्+गीता, śrīmat+bhagavat+gītā)

To focus learning, only the assistant's response is used as the target for the loss function, reinforcing the generation of linguistically accurate decompositions. The constituent morphemes in the target are separated by a + character.

3.2 Fine-Tuning Strategy

To efficiently adapt the model, we use parameter-efficient fine-tuning (PEFT) (Lialin et al., 2023; Ding et al., 2023), specifically Low-Rank Adaptation (LoRA) (Hu et al., 2022b), with 4-bit quantization via the Unsloth framework (Unsloth AI, 2023) to reduce memory usage and mitigate overfitting. We selected a LoRA rank (r) of 32 and scaling factor α=32 after experiments with r=8 (79.6% accuracy), r=16 (82.4%), and r=32 (87.7%) on a validation set demonstrated its superior performance (see Figure 2). The model was trained for one full epoch over 48,000 examples using the AdamW optimizer (Loshchilov and Hutter, 2019) with 0.01 weight decay and a learning rate of 2e-4 with a linear schedule and 5 warmup steps (Hu et al., 2022a). We used a cross-entropy loss on assistant tokens only, a max sequence length of 2048, and an effective batch size of 8 (2 per-device with 4 gradient accumulation steps) to balance stability with memory constraints on A10G GPUs. The full training, implemented in PyTorch (Paszke et al., 2019) and Hugging Face Transformers (Wolf et al., 2020), required approximately 3 GPU hours.

4 Results and Evaluation

4.1 Dataset

For training and evaluation, we utilized a curated dataset derived from the University of Hyderabad (UoH) corpus data (Krishna et al., 2016, 2020), a common resource in prior Sandhi splitting research (Aralikatte et al., 2018; Dave et al., 2021). Our final dataset consists of over 48,000 training examples and a held-out test set of approximately 2,000 examples, with a 10% validation set used for hyperparameter tuning[1]. The data was meticulously prepared for our instruction-tuning approach: each

[1]The actual dataset contains only Devanagari script; transliterations are provided throughout this paper for reader accessibility.

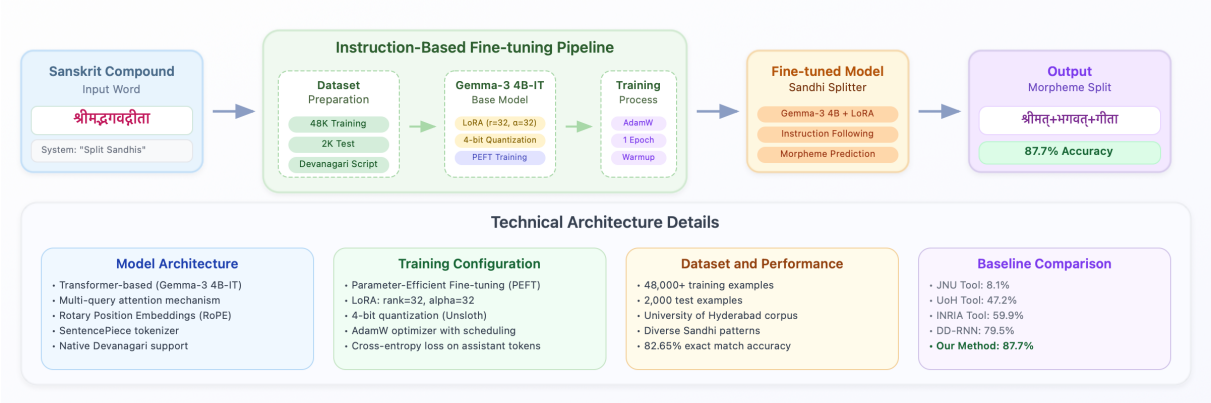

Figure 2: Instruction-Based Fine-tuning Pipeline and Technical Architecture Details for Sanskrit Sandhi Splitting. For detailed discussion of components, see Section 3 and Section 4.

instance pairs a Devanagari compound word (e.g., विद्यालयः (vidyālayaḥ)) with its morphemic split using a + separator (e.g., विद्या+आलयः (vidyā+ālayaḥ)) and is structured in the conversational prompt format described in Section 3. These details are summarized in Figure 2.

4.2 Evaluation Metric

The performance of our fine-tuned Gemma-3 4B model was evaluated using exact match accuracy. This strict metric counts a prediction as correct only if the entire generated sequence of morphemes, including all characters and + separators, perfectly matches the ground truth. We chose this rigorous metric because the Sandhi splitting task demands absolute precision, as partially correct splits are often linguistically invalid and would hinder downstream NLP tasks. This corresponds to the "Split Prediction Accuracy" in our comparative results (Table 1) and is noted in Figure 2.

4.3 Quantitative Results

On the held-out test set of approximately 2,000 examples, our fine-tuned Gemma-3 4B model achieved a **Split Prediction Accuracy (Exact Match) of 87.7%**.

Table 1 provides a detailed comparison of our model's performance against several previously reported systems for Sanskrit Sandhi splitting. This includes traditional rule-based tools (JNU, UoH, INRIA) and specialized neural architectures like the DD-RNN by Aralikatte et al. (2018) and the Two-Stage Seq2Seq model by Dave et al. (2021). The "Location Prediction Accuracy" metric, relevant primarily for models that perform split point detection as a separate stage, is marked as not applicable ("-") for our end-to-end LLM, as it performs the task in a single generative step.

The 87.7% accuracy achieved by our model is a strong result that is highly competitive in this domain. It significantly outperforms traditional tools and surpasses the reported accuracy of specialized architectures like the DD-RNN (79.5%). While the tailored Two-Stage Seq2Seq model by Dave et al. (2021) also achieved a strong accuracy of 86.8%, our approach offers the advantage of a more unified and potentially simpler fine-tuning pipeline. By leveraging a general-purpose pre-trained LLM, we avoid the need to engineer distinct components for location prediction and morpheme generation. This highlights the capability of modern LLMs, adapted through PEFT, to effectively tackle complex, rule-governed linguistic tasks.

4.4 Error Analysis

A qualitative analysis of the 246 incorrect predictions on our 2000-example test set reveals several key limitations. The most common issues were Boundary Errors (35%), where the split location was incorrect (e.g., for input तस्येदम् (tasyedam), the model produced तस्ये+दम् (tasye+dam) instead of the ground truth तस्य+इदम् (tasya+idam)), and Morpheme Reconstruction Errors (28%), with imperfectly restored sounds (e.g., for चिदानन्दः (cidānandaḥ), it produced चिद्+आनन्दः (cid+ānandaḥ) instead of चित्+आनन्दः (cit+ānandaḥ)). Other significant categories included Under-splitting (18%), where a required split was missed (e.g., प्रत्येकम् (pratyekam) was not split into प्रति+एकम् (prati+ekam)), and Over-splitting (12%), where a spurious split was introduced (e.g., अस्ति (asti) was split into अस्+ति (as+ti)). The remaining errors (7%) involved formatting issues or failures on rare Sandhi patterns. This analysis indicates that while the model has learned

Table 1: Comparative Performance on Sanskrit Sandhi Splitting. "Split Prediction Accuracy" refers to exact match accuracy of the final morphemic split. JNU, UoH, and INRIA results are as reported in Dave et al. (2021) from their Table 3, reflecting performance of rule-based/traditional tools on their test sets. DD-RNN results from Aralikatte et al. (2018). Two-Stage Seq2Seq results are from Dave et al. (2021). "Location Prediction Accuracy" is specific to two-stage models.

Model	**Location Prediction Acc (%)**	**Split Prediction Acc (%)**
JNU Tool	-	8.1
UoH Tool	-	47.2
INRIA Tool	-	59.9
DD-RNN (Aralikatte et al., 2018)	95.0	79.5
Two-Stage Seq2Seq (Dave et al., 2021)	92.3	86.8
Gemma-3 4B (Ours)	**-**	**87.7**

many patterns, precise boundary detection in ambiguous contexts, consistent reversal of subtle phonetic changes, and identifying multiple sequential junctions remain key challenges.

4.5 Discussion

The 87.7% exact match accuracy achieved by our fine-tuned Gemma-3 4B model underscores the potential of modern LLMs for specialized linguistic tasks like Sanskrit Sandhi splitting. By combining an instruction-tuning approach (Ouyang et al., 2022) with parameter-efficient fine-tuning (PEFT) methods like LoRA, we effectively adapted the model's extensive pre-trained knowledge, enabling it to implicitly learn complex morpho-phonological rules from data without explicit grammatical encoding.

Our LLM-based approach substantially outperforms traditional rule-based systems and is highly competitive with specialized neural architectures like the DD-RNN (Aralikatte et al., 2018) and the Two-Stage Seq2Seq model (Dave et al., 2021). Notably, our simpler, unified pipeline achieves this strong performance without the architectural complexity of prior multi-component models. This PEFT-facilitated simplification makes advanced NLP more accessible for morphologically complex languages (Tsarfaty et al., 2010; Voutilainen, 1997), a challenge also seen in other Indic languages like Malayalam (DevadathV. et al., 2014; Sebastian and Kumar, 2020), Kannada (Shree et al., 2016), Bangla (Ghosh et al., 2022), and Hindi (Gupta and Goyal, 2009).

However, our error analysis reveals persistent challenges in precise boundary detection for ambiguous splits and the perfect reconstruction of morphemes after subtle phonetic changes. The model's tendency to under- or over-split suggests that refinements like targeted data augmentation or more sophisticated prompting could yield improvements. Despite these limitations, the results are highly encouraging. They demonstrate that fine-tuning moderately-sized LLMs is a viable and efficient strategy for developing robust tools for computational Sanskrit, and the implicit learning paradigm shows promise for other morpho-phonological tasks in classical languages.

5 Conclusion

In this paper, we have presented a data-driven approach for Sanskrit Sandhi splitting by fine-tuning the Gemma-3 4B Large Language Model. Our method leverages parameter-efficient techniques (LoRA) and an instruction-based learning paradigm, achieving a competitive exact match accuracy of 87.7% on a curated dataset of over 50,000 examples. This result demonstrates that general-purpose pre-trained LLMs can be effectively adapted to handle complex, rule-governed morpho-phonological phenomena in Sanskrit without requiring specialized architectures or full model fine-tuning. Our methodology, which combines instruction following with LoRA, offers a scalable and resource-efficient path for tackling similar tasks. The findings affirm the potential of LLMs as powerful and adaptable tools for computational linguistics, especially for morphologically rich and low-resource languages. Future work will focus on refining the instruction-tuning process, exploring more diverse and larger datasets, and investigating methods to integrate lexical or grammatical knowledge to further enhance performance and address the identified error categories.

References

Rahul Aralikatte, Neelamadhav Gantayat, Naveen Panwar, Anush Sankaran, and Senthil Mani. 2018. Sanskrit sandhi splitting using a double decoder rnn. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4909–4914. Association for Computational Linguistics.

Shubham Bhardwaj, Neelamadhav Gantayat, Nikhil Chaturvedi, Rahul Garg, and Sumeet Agarwal. 2018. SandhiKosh: A benchmark corpus for evaluating sanskrit sandhi tools. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Krishnakant Bhatt, N. Karthika, Ganesh Ramakrishnan, and P. Jyothi. 2024. CharSS: Character-level transformer model for sanskrit word segmentation. *arXiv preprint arXiv:2407.06331*, abs/2407.06331.

Rick Briggs. 1985. Knowledge representation in sanskrit and artificial intelligence. *AI magazine*, 6(1):32–39.

George Cardona. 1997. *Pāṇini: a survey of research*. Motilal Banarsidass Publ.

Sushant Dave, Arun Kumar Singh, A. P. Prathosh, and Brejesh Lall. 2021. Neural compound-word (sandhi) generation and splitting in sanskrit language. In *Proceedings of the 8th ACM IKDD CODS and 26th COMAD*.

V. DevadathV., Litton J. Kurisinkel, D. Sharma, and Vasudeva Varma. 2014. A sandhi splitter for malayalam. In *Proceedings of the 11th Intl. Conference on Natural Language Processing (ICON-2014)*, pages 212–218, Goa, India.

Ning Ding, Yujia Qin, Liu Yang, Furu Wei, Zonghan Yang, Yusheng Su, Shengding Li, Pengjun Chen, Yujia Chen, Chi-Min Chen, and 1 others. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Hema Gaikwad and R Jatinderkumar. 2021. On state-of-the-art of POS tagger, 'sandhi' splitter, 'alankaar' finder and 'samaas' finder for indo-aryan and dravidian languages. *International Journal of Advanced Computer Science and Applications*, 12(4).

Neelamadhav Gantayat, Rahul Aralikatte, Naveen Panwar, Anush Sankaran, and Senthil Mani. 2018. Sanskrit sandhi splitting using seq2(seq)2. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4909–4914.

Gemma Team, Google DeepMind, and Google. 2024. Gemma: Open models based on gemini research and technology. Technical report, Google. Accessed on May 31, 2025.

Samarjit Ghosh, Souvik Mukherjee, Debojyoti Roy, S. Sarkar, and Debranjan Sarkar. 2022. Bangla language processing: Sandhi. In *2022 IEEE India Council International Subsections Conference (INDISCON)*, pages 1–5.

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a sanskrit reader. In *Proceedings of the Fifth International Symposium on Sanskrit Computational Linguistics*, pages 130–171, Mumbai, India. D.K. Printworld (P) Ltd.

P. Gupta and Vishal Goyal. 2009. Implementation of rule based algorithm for sandhi-vicheda of compound hindi words. *arXiv preprint arXiv:0909.2379*, abs/0909.2379.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. 2022a. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022b. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

G. Huet. 2003. Towards computational processing of sanskrit. In *International Conference on Universal Knowledge and Language (Icon03)*, Mumbai, India.

Gérard Huet. 2005. A functional toolkit for morphological and phonological processing, application to a sanskrit tagger. *Journal of Functional Programming*, 15(4):573–614.

Paul Kiparsky. 2009. Pāṇini as a variationist. In *Sanskrit computational linguistics: First and Second International Symposia Rocquencourt, France, October 29-31, 2007 Providence, RI, USA, May 15-17, 2008, Revised Selected Papers*, volume 5402 of *Lecture Notes in Computer Science*, pages 1–20. Springer.

Amrith Krishna, Ashim Gupta, Pawan Goyal, Bishal Santra, and Pavankumar Satuluri. 2020. A graph-based framework for structured prediction tasks in sanskrit. *Computational Linguistics*, 46(4):785–845.

Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. 2016. Word segmentation in sanskrit using path constrained random walks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 494–504. The COLING 2016 Organizing Committee.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71. Association for Computational Linguistics.

Amba Kulkarni. 2010. Sanskrit computational linguistics. *Annals of the Bhandarkar Oriental Research Institute*, 91:97–129.

Charles Li. 2023. Using n-aksaras to model sanskrit and sanskrit-adjacent texts. *arXiv preprint arXiv:2301.12969*, abs/2301.12969.

Jingwen Li and Leander Girrbach. 2022. Word segmentation and morphological parsing for sanskrit. *arXiv preprint arXiv:2201.12833*, abs/2201.12833.

Vladislav Lialin, Akim Deshwal, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*. Originally arXiv:1711.05101 [cs.LG] (2017).

Vipul Mittal. 2010. Automatic sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90, Uppsala, Sweden. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Kasmir Raja, R. V, and M. Lakshmanan. 2014. A binary schema and computational algorithms to process vowel-based euphonic conjunctions for word searches. *arXiv preprint arXiv:1409.4354*, abs/1409.4354.

N. Rama and M. Lakshmanan. 2009. A new computational schema for euphonic conjunctions in sanskrit processing. *arXiv preprint arXiv:0911.0894*, abs/0911.0894.

V. Reddy, Amrith Krishna, V. Sharma, Prateek Gupta, R. VineethM., and Pawan Goyal. 2018. Building a word segmenter for sanskrit overnight. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Sreedeepa H S, Ajay K Mani, Arun Kumar C, and S. M. Idicula. Review on sanskrit sandhi splitting using deep learning techniques. *International Journal of Innovative Technology and Developing World*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Ilya Sutskever, Delyan Kubric, Margaret Liu, Lintang Logeswaran, Gaurav Sharma, Manan Dey, and 1 others. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

M. Sebastian and G. Santhosh Kumar. 2020. Machine learning approach to suffix separation on a sandhi rule annotated malayalam data set. *South Asia Research*, 40(2):231–249.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

M. R. Shree, S. Lakshmi, and B. Shambhavi. 2016. A novel approach to sandhi splitting at character level for kannada language. In *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 17–20.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Reut Tsarfaty, Khalil Sima'an, Daniel Gildea, and Joakim Nivre. 2010. Statistical parsing of morphologically rich languages (spmrl): What, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, California. Association for Computational Linguistics.

Unsloth AI. 2023. Unsloth ai github repository. `https://github.com/unslothai/unsloth`. Accessed on May 31, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008.

Atro Voutilainen. 1997. Morphological disambiguation. In *Survey of the State of the Art in Human Language Technology*, pages 105–113. Cambridge University Press for the Commission of the European Communities.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and 1 others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv preprint arXiv:1609.08144*.