BanSuite: A Unified Toolkit and Software Platform for Low-Resource NLP in Bangla

Md. Abu Sayed[1*], Faisal Ahamed Khan[1*], Jannatul Ferdous Tuli[1*], Nabeel Mohammed[3], Mohammad Ruhul Amin[4], Mohammad Mamun Or Rashid[5]

[1]Giga Tech Limited, Dhaka, Bangladesh, [3]North South University, Dhaka, Bangladesh
[4]Fordham University, New York, USA, [5]Bangladesh Computer Council, Dhaka, Bangladesh
Correspondence: faisal.cse06@gigatechltd.com *These authors share first authorship

Abstract

Bangla is one of the world's most widely spoken languages, yet it remains significantly under-resourced in natural language processing (NLP). Existing efforts have focused on isolated tasks such as Part-of-Speech (POS) tagging and Named Entity Recognition (NER), but comprehensive, integrated systems for core NLP tasks including Shallow Parsing and Dependency Parsing are largely absent. To address this gap, we present BanSuite, a unified Bangla NLP ecosystem developed under the EBLICT project. BanSuite combines a large-scale, manually annotated Bangla Treebank with high-quality pretrained models for POS tagging, NER, shallow parsing, and dependency parsing, achieving strong in-domain baseline performance (POS: 90.16 F1, NER: 90.11 F1, SP: 86.92 F1, DP: 90.27 UAS). The system is accessible through a Python toolkit (Bkit) and a Web Application, providing both researchers and non-technical users with robust NLP functionalities, including tokenization, normalization, lemmatization, and syntactic parsing. In benchmarking against existing Bangla NLP tools and multilingual Large Language Models (LLMs), BanSuite demonstrates superior task performance while maintaining high efficiency in resource usage. By offering the first comprehensive, open, and integrated NLP platform for Bangla, BanSuite lays a scalable foundation for research, application development, and further advancement of low-resource language technologies. A demonstration video is provided to illustrate the system's functionality in https://youtu.be/3pcfiUQfCoA

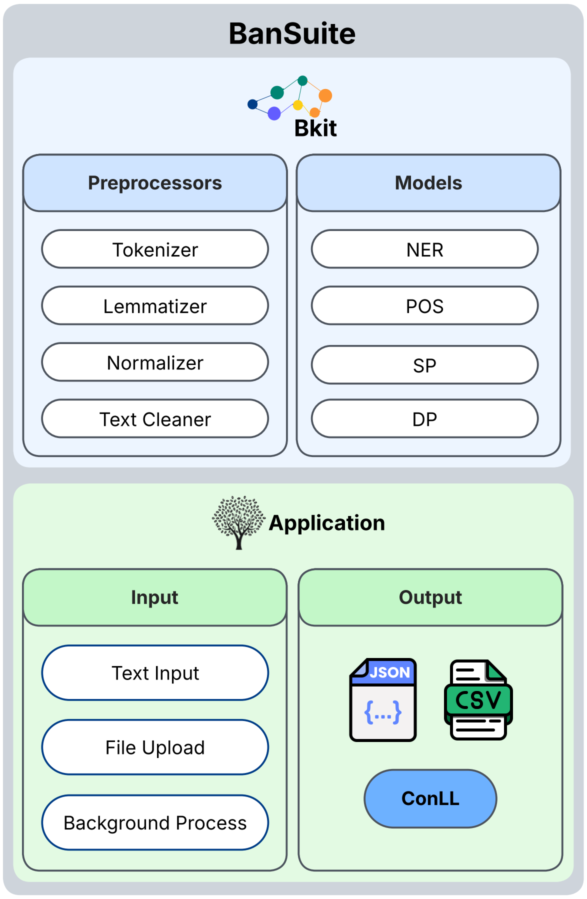


Figure 1: **BanSuite** comprises two complementary components. The **Bkit** package delivers core NLP functionalities. The **Application** provides an interactive NLP prediction platform, enabling users to upload files and obtain results in multiple downloadable formats, thereby facilitating both research and practical usage.

1 Introduction

Bengali, or Bangla, is a classical Indo-Aryan language of the Indo-European family, native to the Bengal region of South Asia (Cardona and Jain, 2007). It has about 242 million native speakers, and 43 million second-language speakers (Eberhard et al., 2024), making it one of the world's most spoken languages (Eberhard et al., 2024). It is the official language of Bangladesh, where around 98% of the population speaks it natively (Eberhard et al., 2024). In India, Bengali is the second most spoken language and serves as an official language in West Bengal, Tripura, and the Barak Valley of Assam. Since 2011, it has also been recognised as the second official language of Jharkhand.

Despite this vast number of speakers, the Bengali language is unfortunately still a low-resource language in Natural Language Processing (NLP). Although some important work on various core components of the Bengali language has been done (Arora, 2020) (Sarker, 2021) sporadically at the research level over the past decade, a fundamental gap has always been evident. Unlike English, which has powerful and integrated ecosystems like Stanford CoreNLP (Manning et al., 2014) or spaCy (Honnibal et al., 2020) that serve as reliable platforms for researchers, academics, and technology enthusiasts interested in the language, a single, open, and integrated platform for Bengali has been absent until now. The Comparison of popular NLP libraries by their Bangla support illustrated in **Table 1**.

We are introducing this integrated Bengali comprehensive Core NLP ecosystem **Figure 1**, built under the Enhancement of Bangla Language in ICT through Research and Development (EBLICT) project, to the academic and research community for the first time. In this article, we have discussed in detail how our system was prepared, what its architecture is, and what core components and resources have been made available for use under this platform.

The process of preparing our system was completed in several steps. First, an annotation platform was created where gold-standard data was annotated by experienced and certified linguists. This annotated data was used to train our core Bengali NLP models, such as Named Entity Recognition (NER), Part-of-Speech (POS) tagging, Shallow Parsing (SP), and Dependency Parsing (DP). The trained models were then deployed into the software system [1] and integrated into the Bangla Text Processing Kit (Bkit) [2]. Bkit is a comprehensive Python library that provides a wide range of functionalities for Bangla NLP, including word tokenization, sentence tokenization, normalization, text cleaning, and lemmatization.

The landscape analysis highlights a significant functional gap in accessible, application-ready advanced systems for the Bangla language. Existing major Bangla toolkits (Sarker, 2021; Arora, 2020) either focus primarily on foundational preprocessing or provide limited support for advanced tasks, without offering high-performance, unified models that are readily usable for complex NLP applications. The BanSuite system is designed to explicitly address this gap by providing robust, dedicated models for resource-intensive, high-value tasks through a consolidated interface.

- We present the first open and unified Bengali core NLP platform, integrating all core NLP components under a single framework.
- Our Pretrained Language Models (PLMs) were trained on a linguist-certified Bengali Treebank. The inter-annotator agreement (IAA) scores, measured using Fleiss' κ (Fleiss, 1971), are: POS: $\kappa = 0.77$, NER: $\kappa = 0.92$, SP: $\kappa = 0.68$, and DP: $\kappa = 0.68$.
- The developed Bangla NLP models achieve n-domain baseline performance scores of POS: 90.16% F1, NER: 90.11% F1, SP: 86.92% F1, and DP: 90.27% UAS on test sets, demonstrating strong baseline results for Bengali NLP tasks.
- We provide access for both technical and non-technical users through two interfaces: the Python API and the Web Application.
- Bridging the low-resource gap creates a foundational ecosystem akin to NLTK, spaCy, or Stanford CoreNLP for Bengali, offering broader core NLP functionalities.

2 Related Work

Early efforts in NLP tool development, exemplified by the Natural Language Toolkit (NLTK) (Bird et al., 2009), provided essential computational interfaces for core linguistic operations such as tokenization, stemming, and access to lexical resources. As the field increasingly shifted toward deep learning, libraries like spaCy (Honnibal et al., 2020) and research-oriented frameworks such as AllenNLP (Gardner et al., 2018) standardized the use of high-performance neural architectures for routine NLP tasks, thereby enabling rapid NLP experimentation.

The Stanford CoreNLP (Manning et al., 2014), widely regarded as a contemporary gold standard for system demonstrations. CoreNLP introduced a unified, language-agnostic, fully neural pipeline for comprehensive text analysis across multiple languages, covering tasks such as parser, tagger. A Python version of CoreNLP, Stanza (Qi et al.,

[1] https://corpus.bangla.gov.bd/corpus/ml-model/tree-bank

[2] https://github.com/eblict-gigatech/bangla-text-processing-kit

Name	**Stanza**	**NLTK**	**AllenNLP**	**Flair**	**SpaCy**	**Stanford CoreNLP**	**iNLTK**	**BNLP**	**BanSuite**
Language	Multilingual (Bangla N/A)	Multilingual (Bangla N/A)	Multilingual (Bangla N/A)	Multilingual (Bangla N/A)	Multilingual (Bangla N/A)	Multilingual (Bangla N/A)	Multilingual (Bangla Supported)	Dedicated Bangla	Dedicated Bangla
Accessibility	Python Package	Python Package	Python Package	Python Package	Python Package	App.	Python Package	Python Package	Python Package + App.

Table 1: Comparison of popular NLP libraries by their Bangla support: dedicated, multilingual, or none (Sarker, 2021; Arora, 2020; Qi et al., 2020; Gardner et al., 2018). Although (Honnibal et al., 2020) lists Bangla support, we found no Bangla-specific functionality. The table also distinguishes toolkit-based libraries from standalone applications.

Feature	**BNLP**	**iNLTK**	**Bkit (Ours)**	**App. (Ours)**
Tokenizer	✓	✓	✓	
Embedding	✓	✓	✓	
Text Cleaning	✓	✓	✓	
Normalizer	✓		✓	
Lemmatizer			✓	
POS	✓		✓	✓
NER	✓		✓	✓
SP			✓	✓
DP			✓	✓
Visualization			✓	✓

Table 2: Comparison of key features in existing Bangla NLP tools (Sarker, 2021; Arora, 2020) and our toolkit, showing coverage of tokenization, embeddings, text cleaning, normalization, lemmatization, and models.

2020), was later developed as a comprehensive library that not only provides core NLP models but also supports fundamental tasks, including tokenization, multi-word token expansion, and lemmatization for multiple languages.

An Indic language-based work is the iNLTK library (Arora, 2020), which aims to lower the barrier for applied research across 13 Indic languages, including Bangla. (Arora, 2020) provides pre-trained models and out-of-the-box support for essential functionalities such as tokenization, word embeddings, textual similarity, and data augmentation. However, (Arora, 2020) does not provide core NLP models and is limited to tasks such as embeddings and sentence similarity.

BNLP (Sarker, 2021) is a dedicated Bangla toolkit that consolidates core preprocessing and tagging capabilities, offering pre-trained models for tokenization, word and document embeddings, POS tagging, and NER. Its primary strength lies in providing essential linguistic analysis for higher-level applications. However, BNLP does not include syntactic parsers, limiting its functionality to foundational tagging tasks. **Table 2** demonstrates the comparison of key features in existing Bangla NLP tools.

3 Development of BanSuite

3.1 Treebank

We use a large-scale, in-house annotated Bangla treebank to train all models presented in this work. The treebank provides multiple layers of linguistic annotation, NER, POS, SP, and DP, and is designed to ensure broad coverage and consistent annotation quality across diverse Bangla text genres. As shown in **Table 3**, the treebank contains roughly 170K sentences and 2M words for each task, along with high inter-annotator agreement (IAA) scores across annotation layers.

Metric	**NER**	**POS**	**SP**	**DP**
Sentence Count	169K	172K	174K	174K
Word Count	2.0M	2.0M	2.0M	2.0M
IAA Score	92.60%	77.30%	68.15	68.54

Table 3: Overall statistics of our Bangla Treebank.

To ensure comprehensive linguistic and domain coverage, the dataset was sourced from a diverse collection of text types, including news, social media, literature, and conversational dialogues. The distribution of these domains across the NER, POS, SP, and DP datasets is illustrated in Figure 2. This diversity enables BanSuite models to generalize effectively across both formal and informal Bangla.

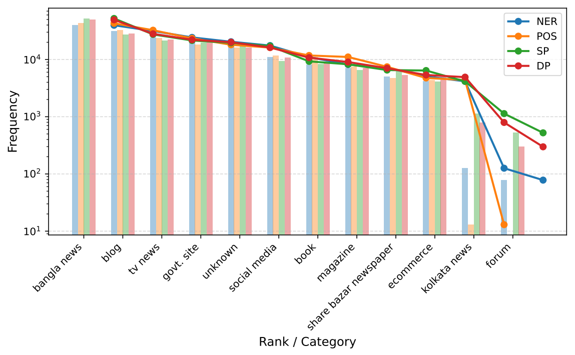


Figure 2: Distribution of source types across NER, POS, SP, and DP datasets. Bars represent absolute frequencies, while lines show log-scale rank–frequency trends.

3.2 Annotation and Validation

Across all four tasks, NER, POS, SP, and DP, we adopted a unified and systematic annotation framework (Islam et al., 2023), as also used by (Mahtab et al., 2025). Each dataset was annotated by trained annotators and subsequently validated by domain experts to ensure accuracy and consistency. Annotations were completed in small groups, where every sentence was independently labeled by three annotators following task-specific instructions. A majority-vote mechanism was used to determine the initial label for each instance. The assigned validator then examined the annotations, resolved inconsistencies, and finalized the labels visualized in **Figure 3**. A more detailed process is available in (Mahtab et al., 2025).

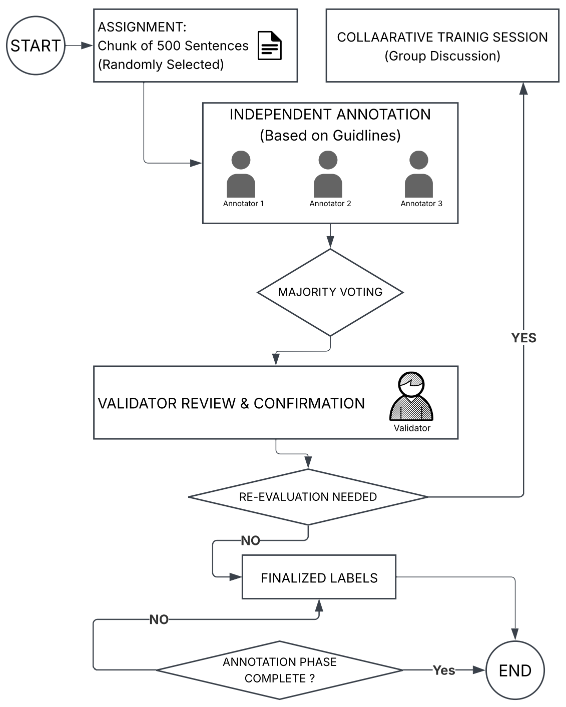

START
ASSIGNMENT: Chunk of 500 Sentences (Randomly Selected)
COLLAARATIVE TRAINIG SESSION (Group Discussion)
INDEPENDENT ANNOTATION (Based on Guidlines)
Annotator 1
Annotator 2
Annotator 3
MAJORITY VOTING
YES
VALIDATOR REVIEW & CONFIRMATION
Validator
RE-EVALUATION NEEDED
NO
FINALIZED LABELS
NO
ANNOTATION PHASE COMPLETE ?
Yes
END

Figure 3: The annotation process involves assigning a chunk of 500 sentences for independent annotation (Mahtab et al., 2025). If re-evaluation is needed, collaborative training sessions are held for group discussions. The process includes majority voting and validator review to confirm labels, which leads to finalized labels or further re-evaluation. This cycle continues until the annotation phase is complete.

The number of annotation groups, assigned instances, validated instances, and acceptance rates are visualized in **Table 4**. These metrics reflect the strong quality-control measures used throughout the dataset construction process.

Metric	NER	POS	SP	DP
Number of Groups	48	147	108	45
Assigned Data	183,201	202,335	179,009	194,604
Annotated Data	180,859	186,791	177,959	192,649
Validated Data	170,261	173,123	174,784	174,415
Rejected Data	10,595	11,015	3,174	13,328
Annotators	86	234	184	95
Validators	7	15	13	13
Acceptance Rate (%)	94.2	92.7	98.2	90.5

Table 4: Annotation workflow statistics for NER, POS, SP, and DP tasks. Acceptance Rate is computed as (Validated / Annotated).

3.3 Models

For part-of-speech tagging and named entity recognition, we adopt the noisy-label learning framework of (Liu et al., 2021), which explicitly models annotation uncertainty through confidence estimation. By applying confidence-aware regularization, the model mitigates the effect of incorrect labels and improves robustness and generalization under imperfect supervision.

For shallow parsing (chunking), we employ the self-attentive constituency parsing architecture proposed in (Kitaev and Klein, 2018) together with its multilingual extension (Kitaev et al., 2019). The use of global contextual representations via self-attention enables more accurate phrase boundary detection while supporting cross-lingual transfer.

For dependency parsing, we utilize a sequence-based generative formulation inspired by (Lin et al., 2022), which linearizes dependency trees into token sequences and predicts them using autoregressive decoding. This formulation simplifies structured prediction while maintaining strong syntactic accuracy and effectively modeling long-distance dependencies.

3.4 Training Hyperparameters

For each task, we selected a suitable pre-trained model based on prior literature and preliminary experiments: (Raffel et al., 2023) for NER, POS, and DP tasks, and (Devlin et al., 2019) for SP. We optimized hyperparameters using a combination of grid search and early stopping. Key hyperparameters tuned included the learning rate, number of epochs, gradient clipping norm, and maximum input sequence length. Gradient clipping with a maximum norm of 1.0 was applied to stabilize training, and gradient accumulation steps were used to simulate larger batch sizes within memory limits. The maximum input sequence length was set to 512 tokens for all tasks. Early stopping based on development set performance was applied to terminate training when no improvement was observed for a pre-defined number of evaluation steps. The final hyperparameter configurations, which yielded the best performance on the development sets, are shown in **Table 5**.

Hyperparameter	NER	POS	SP	DP
Pre-trained Model	T5 Enc	T5 Enc	BERT	T5 Seq2Seq
Learning Rate	1e-5	5e-5	3e-5	5e-5
Epochs	40	40	10	25
Max Grad Norm	1.0	1.0	1.0	1.0
Gradient Accum. Steps	5	5	1	1
Sequence Length	512	512	512	512

Table 5: Hyperparameter configurations used for training the models across all tasks. Values shown are those that achieved the best development-set performance for NER, POS, SP, and DP

3.5 Bkit: Python Package

Bkit is a unified Python package for Bangla NLP that integrates preprocessing and model inference via a modular, consistent API. It supports text cleaning, tokenization, normalization, and lemmatization (Afrin et al., 2023), producing consistent lemmas while handling orthographic variability. Models can be initialized with a single class, automatically leveraging CPU/GPU resources. Built-in visualization tools facilitate interpretation of syntactic and semantic structures, especially for shallow and dependency parsing (Figure 4). Usage examples and API details are provided in **Appendix B**.

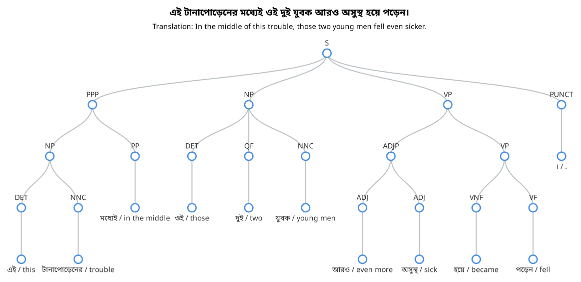


Figure 4: Shallow parsing visualization of a Bengali sentence. Nodes represent syntactic categories, leaf nodes correspond to words, and edges indicate parent–child relationships.

3.6 Web Application

BanSuite provides a web-based inference interface **Figure 5**. Users can perform real-time single-sentence inference or asynchronous batch processing via CSV uploads. Requests are managed through an API Gateway, routing real-time tasks to a synchronous service and batch jobs to worker pools, with results stored for access. Unlike traditional tools like Stanford CoreNLP (Manning et al., 2014), BanSuite requires no local setup and is accessible online up to a usage limit. Further application details are in **Appendix A**.

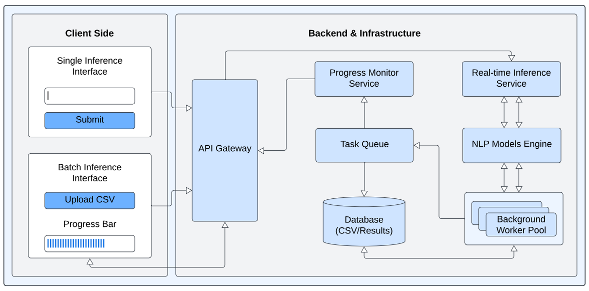


Figure 5: Overview The Inference Service through application, Task Queue, and Background Worker, with core processing in the Models Engine and results stored in a Database for easy access.

4 Result Analysis

4.1 In-domain Baseline Performance

Table 6 reports the in-domain performance of our NER, POS, SP, and DP models. The system achieves strong and consistent results, with development scores ranging from 86–91% and test set performance closely aligned. These results provide a reliable upper bound under matched training and evaluation conditions and serve as a reference for assessing generalization in zero-shot scenarios.

Dataset	NER	POS	SP	DP
Dev Set (%)	91.44	89.91	86.19	90.68
Test Set (%)	91.11	91.16	86.92	90.27

Table 6: In-domain performance of our models on the gold dataset.

4.2 Zero-shot Evaluation

We evaluate BanSuite in a zero-shot setting across four core Bengali NLP tasks: NER, POS, SP, and DP. Performance is compared against open-source Bangla toolkits (Sarker, 2021) and state-of-the-art multilingual LLMs, including Gemini 2.5F (Google DeepMind, 2025), Gemma 3 (27B) (Team et al., 2025), GPT-OSS-20B (OpenAI et al., 2025), and Llama 4 Scout (Meta AI, 2025). To explore cost-efficient alternatives, we also evaluate locally runnable variants of Gemma3[3],

[3]https://ollama.com/library/gemma3:27b

Llama4[4], and GPT-OSS[5] via Ollama, which supports efficient execution and KV-caching (Pope et al., 2022).

Table 7 reports zero-shot results (F1 for NER, POS, SP; UAS for DP). BanSuite consistently achieves the highest scores: 88.78 F1 on NER, substantially outperforming Gemini 2.5F (56.29), Gemma 3 (35.59), and Llama 4 Scout (27.05). On POS, it averages 79.93 F1, ahead of BNLP (50.94–55.10), GPT-OSS-20B (64.01), and other open models. For syntactic tasks, BanSuite leads with 52.56 F1 on SP and 72.19 UAS on DP, while Gemini 2.5F and GPT-OSS-20B achieve moderate scores, and Llama 4 Scout trails. BNLP provides a lightweight competitive baseline, but BanSuite establishes a clear upper bound on zero-shot performance across Bangla NLP tasks.

4.2.1 Few-Shot Analysis

LLMs are known to produce different outputs depending on the prompt (Zhuo et al., 2024). To study this effect, we evaluate Gemma3 27B under both zero-shot and few-shot settings. **Figure 6** shows that providing a few examples in the prompt improves performance by $\approx$ 11.37% across all tasks. These results highlight that carefully designed prompts can influence the LLM's behavior across different tasks.

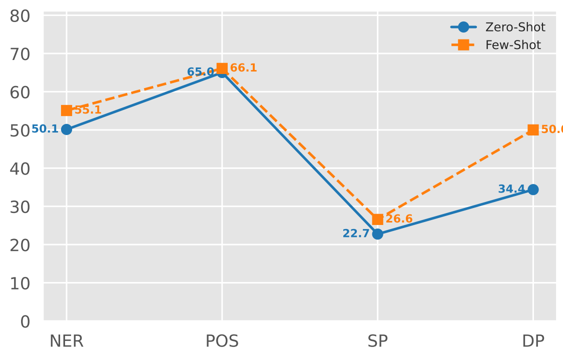


Figure 6: Comparison of Gemma 3 27B performance in zero-shot and few-shot settings across four tasks. Adding 3 examples improves NER and SP performance, has a limited impact on POS, and big improvement in DP performance.

4.3 Inference Cost Analysis

The trade-offs between model performance and power consumption across the evaluated systems are shown in **Figure 7**. Large models such as Llama 4 Scout ($\approx$ 41%), Gemma 3 27B ($\approx$ 43%), and GPT-OSS 20B ($\approx$ 51%) achieve moderate task performance but require over 200 W on average, reflecting substantial energy demands. In contrast, BanSuite (T5) delivers 72% accuracy while consuming under 100 W, demonstrating a favorable balance between performance and efficiency, see **Appendix C**

At the low-power extreme, BNLP (Lafferty et al., 2001) runs entirely on CPU using only $\approx$ 30 W, though its task performance is lower at $\approx$ 27%. Overall, this comparison highlights how model design influences both energy efficiency and effectiveness, allowing informed choices depending on deployment constraints.

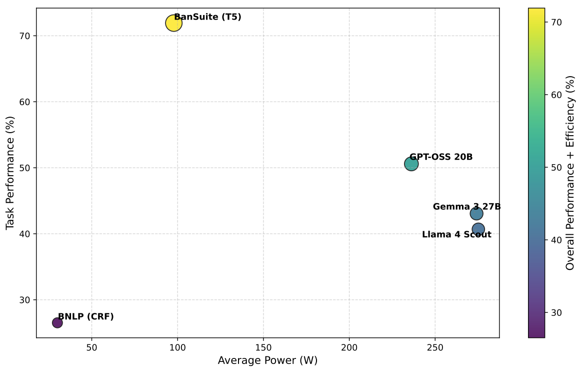


Figure 7: Trade-off between power usage and task performance across models. Point size and color indicate efficiency, with larger and darker points representing higher efficiency.

Limitations

While BanSuite provides a unified toolkit and platform for Bangla NLP, several limitations remain. First, the system relies primarily on in-house annotated data, which, despite their size and diversity, may not capture all linguistic variations, dialects, or domain-specific usage in real-world Bangla text. Second, the current models are trained on standard and semi-formal text, so performance on highly informal or code-mixed data may be lower than reported benchmark results. Third, some complex linguistic phenomena, such as long-range dependencies in dependency parsing or nested entities in NER, remain challenging due to sequence length and model architecture limitations. Fourth, to evaluate existing models like (Sarker, 2021) for POS and NER, we mapped their data tags to our annotation scheme, which can introduce slight inconsistencies or misalignments. In contrast, no tag mapping was applied when evaluating LLMs, which

[4]https://ollama.com/library/llama4:16x17b
[5]https://ollama.com/library/gpt-oss:20b

Tasks	Datasets	BNLP	Llama4 Scout	Gemma3 27B	GPT-OSS-20B	Gemini 2.5F	BanSuite
NER	(Mahtab et al., 2025)	52.65	27.05	35.59	40.64	56.29	88.78
	(Haque et al., 2023)	66.14	50.76	50.53	53.86	54.76	79.80
	(Mhaske et al., 2023)	46.50	65.86	66.25	73.60	77.65	82.50
	Average	55.10	47.22	50.12	55.97	60.20	**83.03**
POS	(Bali et al., 2010)	52.51	55.01	64.93	63.03	72.50	79.95
	(Kharagpur, 2005)	49.36	53.04	65.10	65.00	72.93	79.90
	Average	50.94	54.03	65.02	64.01	72.72	**79.93**
SP	(Mishra et al., 2024)	–	17.03	22.74	28.53	29.81	**52.56**
DP	(Jannat et al., 2021)	–	44.44	34.38	53.85	57.69	**72.19**

Table 7: Zero-shot evaluation (%) of available Bangla unified toolkit and LLMs across different tasks. F1 scores are reported for NER, POS, and SP tasks, while UAS is reported for DP. Scores are reported on multiple benchmark datasets, with averages computed where applicable. A dash (-) indicates that the feature is not available for the corresponding model. BanSuite consistently achieves the highest performance across all tasks.

may limit the comparability of their performance against BanSuite and (Sarker, 2021). Moreover, LLM performance can often be improved with carefully designed prompts (see **Section 4.2.1**), suggesting that their reported results may not fully reflect their potential with optimized prompting strategies. Finally, BanSuite currently focuses exclusively on Bangla and does not provide support like (Manning et al., 2014; Qi et al., 2020; Honnibal et al., 2020) for other low-resource languages, which limits its applicability to multilingual NLP research or cross-lingual tasks.

Future work could focus on expanding annotated corpora, improving support for code-mixed and informal text, refining tag mapping procedures, and adding built-in deployment features to Bkit for effortless deployment.

Acknowledgments

We sincerely thank the *Bangla Syntactic Treebank Corpus with Processing Pipeline and Distribution Platform* project, part of the Enhancement of Bangla Language in ICT through Research and Development (EBLICT)[6], supported by the Bangladesh Computer Council[7] under the ICT Division of the Government of Bangladesh[8].

Our gratitude extends to the development consultant team at Giga Tech Limited[9] and the Department of Communication Disorders, University of Dhaka[10] for their essential annotation support.

We also thank the Institute of Information Technology (IIT), University of Dhaka, for their valuable contribution as the testing team.

We gratefully acknowledge the financial support from the People's Republic of Bangladesh, which enabled this research, and we thank the support teams at Facebook[11] and YouTube[12] for their assistance with accessibility and content integration.

References

Sadia Afrin, Md. Shahad Mahmud Chowdhury, Md. Islam, Faisal Khan, Labib Chowdhury, Md. Mahtab, Nazifa Chowdhury, Massud Forkan, Neelima Kundu, Hakim Arif, Mohammad Mamun Or Rashid, Mohammad Amin, and Nabeel Mohammed. 2023. BanLemma: A word formation dependent rule and dictionary based Bangla lemmatizer. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3695–3710, Singapore. Association for Computational Linguistics.

Gaurav Arora. 2020. iNLTK: Natural language toolkit for indic languages. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 66–71, Online. Association for Computational Linguistics.

Kalika Bali, Monojit Choudhury, and Priyanka Biswas. 2010. Indian language part-of-speech tagset: Bengali (ldc2010t16). Linguistic Data Consortium, Philadelphia. https://catalog.ldc.upenn.edu/LDC2010T16. Web download.

[6]http://eblict.gov.bd/
[7]https://bcc.gov.bd/
[8]https://ictd.gov.bd/
[9]https://gigatechltd.com/
[10]https://www.du.ac.bd/body/LIN
[11]https://www.facebook.com/help/1020633957973118
[12]https://support.google.com/youtube/answer/9783148

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

George Cardona and Dhanesh Jain. 2007. *The Indo-Aryan Languages*. Routledge.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

David M Eberhard, Gary F Simons, and Charles D Fennig. 2024. *Ethnologue: Languages of the World*, 27th edition. SIL International, Dallas, Texas.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Google DeepMind. 2025. Gemini 2.5 flash. https://ai.google.dev/gemini-api/. Accessed: 2025-11-21.

Md. Zahidul Haque, Sakib Zaman, Jillur Rahman Saurav, Summit Haque, Md. Saiful Islam, and Mohammad Ruhul Amin. 2023. B-ner: A novel bangla named entity recognition dataset with largest entities and its baseline evaluation. *IEEE Access*, 11:45194–45205.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python. https://spacy.io.

Md. Ekramul Islam, Labib Chowdhury, Faisal Ahamed Khan, Shazzad Hossain, Md Sourave Hossain, Mohammad Mamun Or Rashid, Nabeel Mohammed, and Mohammad Ruhul Amin. 2023. Sentigold: A large bangla gold standard multi-domain sentiment analysis dataset and its evaluation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 4207–4218, New York, NY, USA. Association for Computing Machinery.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Siratun Jannat, Mizanur Rahoman, Shafi Sourov, Jannatul Ferdaousi, Syeda Shahzadi, and Daniel Zeman. 2021. Ud bengali bru (universal dependencies v2.9). https://github.com/UniversalDependencies/UD_BengaliâĂŚBRU. Accessed: 2025-11-19; License: CCBY-SA4.0.

IIT Kharagpur. 2005. Indian language pos-tagged corpus. https://www.nltk.org/api/nltk.corpus.reader.indian.html. Accessed: 2025-11-11.

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

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Boda Lin, Zijun Yao, Jiaxin Shi, Shulin Cao, Binghao Tang, Si Li, Yong Luo, Juanzi Li, and Lei Hou. 2022. Dependency parsing via sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7339–7353, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kun Liu, Yao Fu, Chuanqi Tan, Mosha Chen, Ningyu Zhang, Songfang Huang, and Sheng Gao. 2021. Noisy-labeled NER with confidence estimation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3437–3445, Online. Association for Computational Linguistics.

Md. Motahar Mahtab, Faisal Ahamed Khan, Md. Ekramul Islam, Md. Shahad Mahmud Chowdhury, Labib Imam Chowdhury, Sadia Afrin, Hazrat Ali, Mohammad Mamun Or Rashid, Nabeel Mohammed, and Mohammad Ruhul Amin. 2025. BanNERD: A benchmark dataset and context-driven approach for Bangla named entity recognition. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6807–6828, Albuquerque, New Mexico. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Meta AI. 2025. Llama 4 scout: A natively multimodal mixture-of-experts model. Version 17B-16E, released April 5, 2025.

Arnav Mhaske, Harshit Kedia, Sumanth Doddapaneni, Mitesh M. Khapra, Pratyush Kumar, Rudra Murthy, and Anoop Kunchukuttan. 2023. Naamapadam: A large-scale named entity annotated data for Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10441–10456, Toronto, Canada. Association for Computational Linguistics.

Pruthwik Mishra, Vandan Mujadia, and Dipti Misra Sharma. 2024. Multi task learning based shallow parsing for indian languages. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 23(9).

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. *Preprint*, arXiv:2211.05102.

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*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From words to watts: Benchmarking the energy costs of large language model inference. *Preprint*, arXiv:2310.03003.

Sagor Sarker. 2021. Bnlp: Natural language processing toolkit for bengali language. *Preprint*, arXiv:2102.00405.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

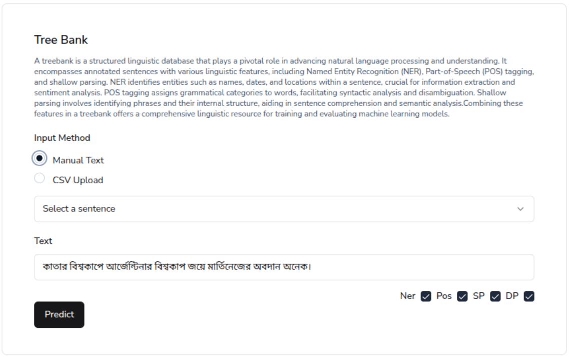
Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and understanding the prompt sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976, Miami, Florida, USA. Association for Computational Linguistics.

Appendix

A Web Application

A.1 Single Inference

BanSuite provides an interactive web interface that allows users to perform single-sentence inference directly from a text input field. The interface supports real-time model execution and immediate visualization of results, making it suitable for quick experimentation and model inspection. See **Figure 8**.



Tree Bank

A treebank is a structured linguistic database that plays a pivotal role in advancing natural language processing and understanding. It encompasses annotated sentences with various linguistic features, including Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and shallow parsing. NER identifies entities such as names, dates, and locations within a sentence, crucial for information extraction and sentiment analysis. POS tagging assigns grammatical categories to words, facilitating syntactic analysis and disambiguation. Shallow parsing involves identifying phrases and their internal structure, aiding in sentence comprehension and semantic analysis.Combining these features in a treebank offers a comprehensive linguistic resource for training and evaluating machine learning models.

Input Method

● Manual Text
○ CSV Upload

Select a sentence

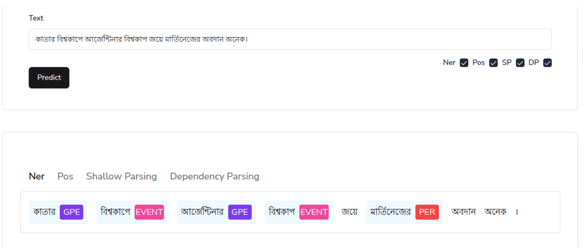
Text

কাতার বিশ্বকাপে আর্জেন্টিনার বিশ্বকাপ জয়ে মার্তিনেজের অবদান অনেক।

Ner ✓ Pos ✓ SP ✓ DP ✓

Figure 8: Web-based interface for single-sentence inference.

Named Entity Recognition After processing the input sentence, the output is presented in a highly intuitive format where the original sentence is displayed, and each recognized entity is visually tagged and highlighted directly within the text. For example, a geographical location might be tagged as GPE (Geopolitical Entity), an event as EVENT, and a person's name as PER. See **Figure 9**.



Text

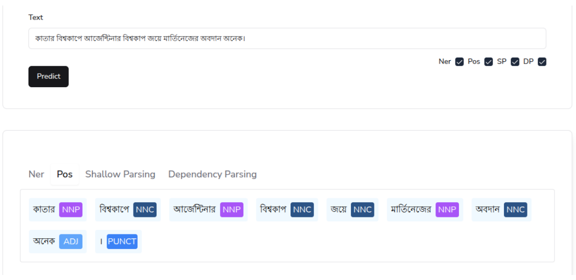
কাতার বিশ্বকাপে আর্জেন্টিনার বিশ্বকাপ জয়ে মার্তিনেজের অবদান অনেক।

Ner ✓ Pos ✓ SP ✓ DP ✓

কাতার GPE বিশ্বকাপে EVENT আর্জেন্টিনার GPE বিশ্বকাপ EVENT জয়ে মার্তিনেজের PER অবদান অনেক ।

Figure 9: NER analysis on a sample Bangla sentence within the Tree Bank module: the output highlights each identified entity with its assigned tag

Part-of-Speech tagging, which assigns grammatical categories to every word in the input sentence, such as noun, verb, adjective, or determiner. The resulting structured output is fundamental for facilitating deeper linguistic analysis, including syntactic analysis and word sense disambiguation. See **Figure 10**.


Text

কাতার বিশ্বকাপে আর্জেন্টিনার বিশ্বকাপ জয়ে মার্তিনেজের অবদান অনেক।

Ner ✓ Pos ✓ SP ✓ DP ✓

Figure 10: PoS tagging output showing each word in the sentence classified by its grammatical function

Shallow Parsing can also be done on the input text. This intermediate level of syntactic analysis aims to identify and segment the sentence into non-overlapping phrases or chunks. This process is instrumental in identifying relationships between major elements in a sentence and aids in both sentence comprehension and more complex downstream tasks. See **Figure 11**.

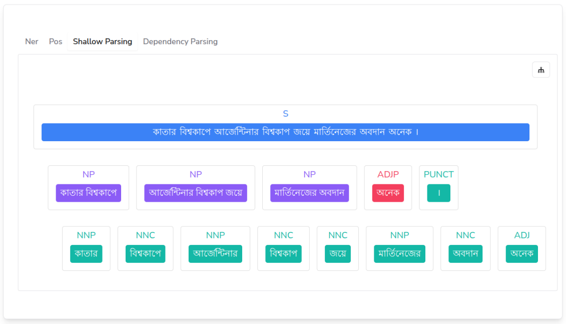
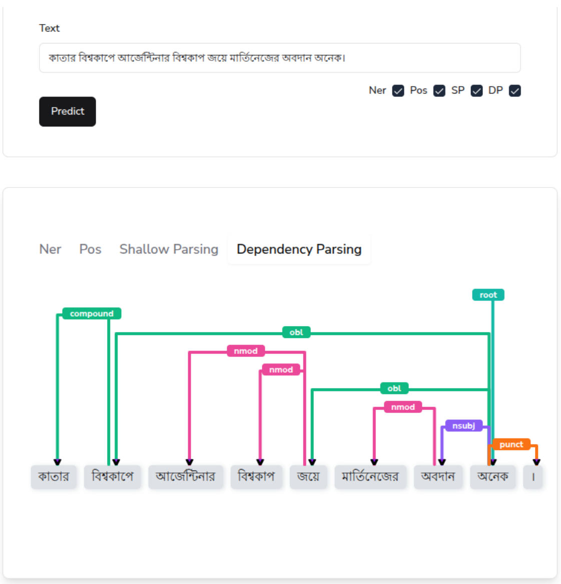


Figure 11: Visualization of Shallow Parsing where the recognized chunks, such as noun phrases (NP) and Adjective Phrases (ADJP), are marked right above the sequence of PoS-tagged words.

Dependency Parsing is the last layer of linguistic analysis provided by the treebank application. This method examines the grammatical links between words in a sentence. It frames the phrase as a directed graph, where each word is a node, and the relationship is represented by a named, directed arc from a head word to its dependent word. See **Figure 12**.


Text

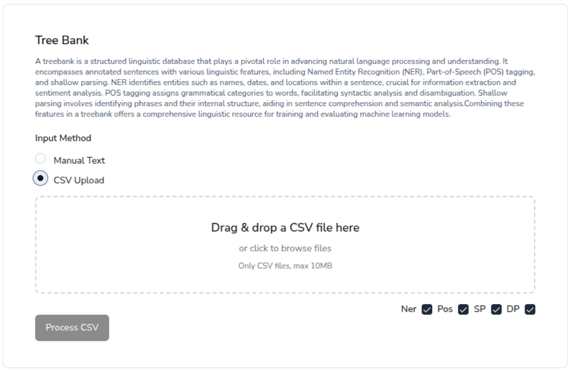
কাতার বিশ্বকাপে আর্জেন্টিনার বিশ্বকাপ জয়ে মার্তিনেজের অবদান অনেক।

Ner ✓ Pos ✓ SP ✓ DP ✓

Figure 12: Visualization of a Dependency Parsing tree where the arcs are connecting the words, along with the specific relationship labels

A.2 Batch Inference

For large-scale processing, the BanSuite application supports batch inference through file uploads. Users can submit CSV files containing multiple input samples, which are processed asynchronously in the background. This feature enables the efficient handling of extensive datasets without requiring manual repetition. See **Figure 13**.


Tree Bank

A treebank is a structured linguistic database that plays a pivotal role in advancing natural language processing and understanding. It encompasses annotated sentences with various linguistic features, including Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and shallow parsing. NER identifies entities such as names, dates, and locations within a sentence, crucial for information extraction and sentiment analysis. POS tagging assigns grammatical categories to words, facilitating syntactic analysis and disambiguation. Shallow parsing involves identifying phrases and their internal structure, aiding in sentence comprehension and semantic analysis.Combining these features in a treebank offers a comprehensive linguistic resource for training and evaluating machine learning models.

Input Method

Manual Text [✓ CSV Upload]

Drag & drop a CSV file here

or click to browse files

Only CSV files, max 10MB

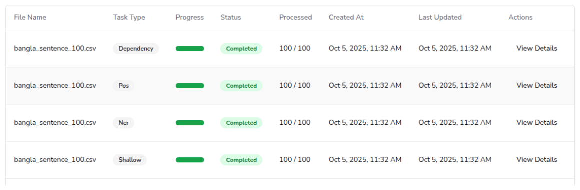
Ner ✓ Pos ✓ SP ✓ DP ✓

Figure 13: Option to upload a CSV file for batch inference.

During processing, the interface provides live progress feedback, allowing users to monitor ongoing inference tasks and retrieve results once computation is complete. See **Figure 14**.

B Bkit

Bkit exposes a unified Python package that integrates preprocessing and model inference under a consistent API. It enables researchers to compose reproducible Bangla NLP pipelines with minimal



File Name	Task Type	Progress	Status	Processed	Created At	Last Updated	Actions
bangla_sentence_100.csv	Dependency	🟩	Completed	100 / 100	Oct 5, 2025, 11:32 AM	Oct 5, 2025, 11:32 AM	View Details
bangla_sentence_100.csv	Pos	🟩	Completed	100 / 100	Oct 5, 2025, 11:32 AM	Oct 5, 2025, 11:32 AM	View Details
bangla_sentence_100.csv	Ner	🟩	Completed	100 / 100	Oct 5, 2025, 11:32 AM	Oct 5, 2025, 11:32 AM	View Details
bangla_sentence_100.csv	Shallow	🟩	Completed	100 / 100	Oct 5, 2025, 11:32 AM	Oct 5, 2025, 11:32 AM	View Details

Figure 14: Dashboard view of uploaded file being processed in the background.

code while ensuring explicit configuration and deterministic behavior.

B.1 Text Cleaning

The cleaning module standardizes text by removing unwanted symbols and normalizing Unicode. It supports configurable filters, allowing transparent preprocessing for reproducibility.

```
import bkit
bkit.transform.clean_text(
    text,
    remove_punctuations=True,
    remove_digits=True,
    remove_emojis=True,
    remove_non_bangla=True,
)
```

B.2 Tokenizer

The tokenizer applies Bangla-specific rules and supports rule-based segmentation. It returns token spans and offsets for precise mapping to model outputs.

```
import bkit
text = "Some Bangla text here"
tokens = bkit.tokenizer.tokenize(text)
```

B.3 Normalizer

Normalization mitigates orthographic variability in Bangla by harmonizing visually similar characters, diacritics, and spacing conventions.

```
import bkit
normalizer = bkit.transform.Normalizer(
    normalize_characters=True,
    normalize_zw_characters=True,
    normalize_halant=True,
    normalize_vowel_kar=True,
    normalize_punctuation_spaces=True
)
normalizer(text)
```

B.4 Lemmatizer

The lemmatizer combines a rule- and dictionary-based system (Afrin et al., 2023) with lexical lookup to generate consistent lemmas while preserving meaning-critical morphemes.

```
import bkit
lemmatized = bkit.lemmatizer.lemmatize(text)
```

B.5 Inference Pipeline

The API is designed to be both simple and user-friendly. By importing a single class, users can initialize the model without additional configuration.

```
from bkit.ner import Infer
model = Infer('ner-noisy-label')
predictions = model(text)
```

```
from bkit.pos import Infer
model = Infer('pos-noisy-label')
predictions = model(text)
```

```
from bkit.shallow import Infer
model = Infer('pos-noisy-label')
predictions = model(text)
```

```
from bkit.dependency import Infer
model = Infer('dependency-parsing')
predictions = model(text)
```

C Inference Cost Analysis

LLMs impose substantial computational demands during inference (Samsi et al., 2023), making efficiency a critical factor for deployment in low-resource or real-time scenarios. To quantify these costs, we measured power consumption and GPU utilization for four model families: Gemma 3 27B, Llama 4 Scout, GPT-OSS, and our proposed BanSuite (T5) model running identical inference workloads on 2$\times$ Nvidia A40 GPUs.

As shown in **Figure 15**, Gemma 3 and Llama 4 Scout exhibit the highest energy requirements, averaging 274.14 W and 275.18 W, respectively. GPT-OSS also maintains a substantial draw at 236.18 W. In contrast, BanSuite operates at just 97.81 W, reducing power consumption by a factor of $\approx 2.4\times$ relative to GPT-OSS and $\approx 2.8\times$ relative to Gemma 3 and Llama 4 Scout.

Figure 16 illustrates the corresponding utilization utilization. Gemma 3 drives the GPUs to consistently high load, averaging 84.37% utilization with brief synchronization-related dips. Llama 4 Scout (Meta AI, 2025) shows considerably lower but more variable usage, averaging 28.48%, reflecting the intermittent expert activation characteristic

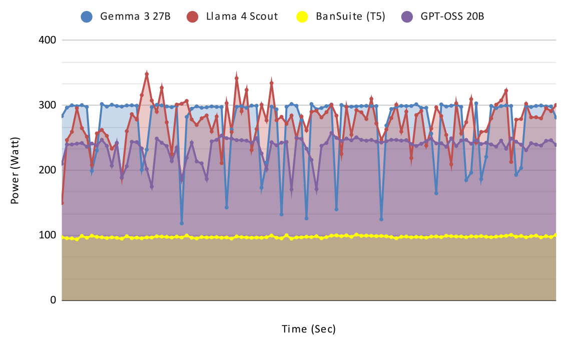


Figure 15: Power consumption of different models during NER inference. Gemma 3 and Llama 4 Scout have the highest usage, GPT-OSS is moderate, and BanSuite is the most efficient.

of Mixture of Experts (MoE) architectures (Jacobs et al., 1991). GPT-OSS 20B (OpenAI et al., 2025), which also follows this active expert (Jacobs et al., 1991), displays similar utilization at 76.24%.

BanSuite, despite its competitive performance, maintains a stable and low utilization profile, averaging 23.50%. This smooth utilization trace mirrors its compact 0.24B parameter footprint and efficient T5 (Raffel et al., 2023).

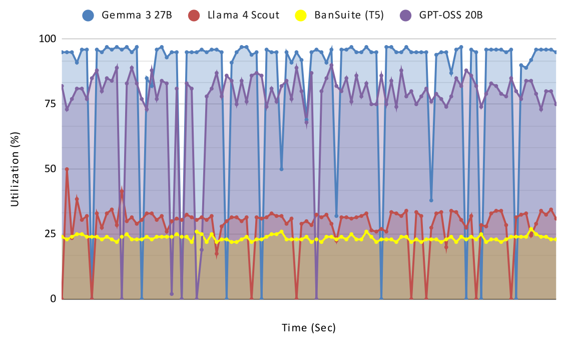


Figure 16: GPU utilization over time for Gemma 3 27B, Llama 4 Scout, and BanSuite (T5). Gemma 3 maintains consistently high load with brief dips, Llama 4 Scout shows moderate but variable usage due to its MoE architecture, and BanSuite (T5) exhibits stable low utilization, reflecting its compact and efficient design.