Using AI to Better Segment Complex Scripts

Mentor: Younies, Shane, Sahand Contributor:Anushka

Abstract

Many East and South-East Asian languages lack clear word boundaries, making text segmentation challenging. Current Unicode models (Dictionary, LSTM, AdaBoost) have limitations, especially for digitally disadvantaged languages (DDLs). This project will optimize the existing LSTM-based segmentation model through quantization, pruning, and hyperparameter tuning, reducing model size by **30-40%** while improving inference speed by **20-25%**. Additionally, a **lightweight Transformer-based model (DistilBERT/ALBERT)** will be developed and compared against LSTM. A reproducible test suite will measure improvements across **Japanese, Thai, and Cantonese**. If successful, the optimized model will be integrated into **Unicode's ICU4X (Rust)** for real-world application.

Project Goals

This project aims to **improve the Unicode word segmentation models** by optimizing existing machine learning approaches and developing new segmentation techniques. The primary focus is to enhance **accuracy, model size, and inference speed** while maintaining compatibility with internationalization (i18n) requirements.

Develop a reproducible test suite to systematically evaluate segmentation models based on accuracy, speed, and model size across CJK, SEA, and DDL languages.

Benchmark and analyze existing Unicode models (Dictionary, LSTM, AdaBoost) to identify their strengths and weaknesses.

Optimize LSTM-based segmentation models using **quantization, pruning, and hyperparameter tuning**, aiming to reduce model size by **30-40%** while maintaining segmentation accuracy.

Train a lightweight Transformer-based segmentation model (DistilBERT/ALBERT) and compare its performance against LSTM-based approaches.

Enhance segmentation for underrepresented scripts by improving dataset diversity, particularly for **Cantonese, Mon, and Shan languages**.

Integrate the improved model into Unicode's ICU4X (Rust) if it outperforms existing approaches in segmentation efficiency.

Implementation

I plan to approach the implementation in 4 phases. This project focuses on optimizing **Unicode's LSTM-based segmentation model** to improve **accuracy, model size, and inference speed**. Additionally, I will be exploring a **Transformer-based model (DistilBERT/ALBERT)** as an alternative. The implementation will follow a four-phase approach with the final two weeks allocated for buffer time.

Phase 1 - Model Benchmarking & Test Suite Development

Benchmark Existing Models – Evaluating Unicode's **Dictionary, LSTM, and AdaBoost** models on accuracy, speed, and model size across **Japanese (CJK), Thai (SEA), and Cantonese (DDL)** datasets.
Develop a Test Suite – Standardize evaluation metrics for segmentation models.
Baseline Performance Analysis – Identifying inefficiencies in current models.
Expected Impact: Establish clear benchmarks for comparison before optimization.

Phase 2 - LSTM Model Optimization

Quantization & Pruning – Reduce model size **by 30-40%** while maintaining accuracy loss **≤2%**, improving inference speed **by 20-25%**.
Hyperparameter Tuning – Adjust dropout, batch size, and learning rate for **5-8% accuracy improvement**.
BiLSTM Integration – Improve segmentation quality by **5-7%** with bidirectional processing.
Dataset Expansion – Augment data for **digitally disadvantaged languages (DDL)**.
Expected Impact: A **smaller, faster LSTM** with improved segmentation accuracy.

Phase 3 - Transformer-Based Model Exploration

Train a Lightweight Transformer (DistilBERT/ALBERT) – Compare performance against optimized LSTM.
Use SentencePiece Tokenization – Improve segmentation for mixed-script languages.
Evaluate Trade-offs – Assess **accuracy, speed, and model size** differences between LSTM and Transformer models.
Expected Impact: Identify whether Transformers outperform LSTM-based segmentation for Unicode.

Phase 4 - Final Optimization & Integration

Compare Best Performing Models – Choose the most efficient model based on benchmarks.
Final Refinements – Further optimize accuracy, reduce size, and improve efficiency.
ICU4X Integration (Stretch Goal) – If successful, begin Rust implementation for Unicode's segmentation pipeline.
Expected Impact: Deliver an optimized segmentation model ready for real-world deployment

Final Buffer Period

Performance Validation- Rerun all benchmarking tests and verify segmentation accuracy across multiple datasets.
Error Handling & Debugging-Identify and resolve any remaining issues in segmentation logic or model deployment.
Final Documentation- Ensure detailed documentation of implementation, datasets, and test results.
Mentor Feedback & Refinement -Incorporate final suggestions from mentors and the Unicode community.
Submission & Review-Prepare and submit the final GSoC report and code contributions.

Estimated Impact

Optimization	Accuracy	Size Reduction	Speed Improvement
Quantization & Pruning	≤2% loss	**30-40% smaller**	**20-25% faster**
Hyperparameter Tuning	**+5-8%**	No change	**10-15% faster**
BiLSTM Integration	**+5-7%**	Slight increase	Improved segmentation
Transformer Model (BERT)	Comparable	More memory-efficient	Faster inference

Technical details

LSTM Optimization-Apply **quantization, pruning, and hyperparameter tuning** to reduce model size (**30-40%**) while maintaining segmentation accuracy (**±2% loss max**).

BiLSTM Integration: Improve contextual segmentation accuracy (**5-7% improvement**).
Transformer Model Exploration-Train a **lightweight Transformer (DistilBERT/ALBERT)** for segmentation and compare with LSTM models.

SentencePiece Tokenization-Enhance segmentation of **complex scripts** and mixed-language text.
Benchmarking & Testing - Develop a **test suite** evaluating **accuracy (F1-score), model size, and inference speed** across CJK, SEA, and DDL languages.

ICU4X Integration (Stretch Goal)- Modify **Rust-based Unicode segmentation logic** to implement improvements.

Challenges & Mitigation

Accuracy Loss from Pruning & Quantization

- Risk: Reducing model size by 30-40% may lead to accuracy degradation beyond the targeted ≤2% loss.
- Mitigation
    - Will apply gradual pruning and structured sparsity instead of aggressive weight removal.
    - Use mixed-precision quantization, reducing precision only in non-critical layers.
    - Conduct iterative benchmarking to ensure minimal trade-off between size reduction and accuracy.

BiLSTM May Increase Model Size Excessively

- Risk: While BiLSTM can improve segmentation by 5-7%, it may also increase model size beyond acceptable limits.
- Mitigation
    - will Implement layer-wise pruning to compress the model without affecting bidirectional context learning.
    - Use knowledge distillation to transfer improvements into a smaller model.
    - Continuously benchmark the trade-off between accuracy gains vs. size increase.

Transformer-Based Model May Not Outperform LSTM

- Risk: DistilBERT/ALBERT may require higher computational resources and not significantly outperform an optimized LSTM.
- Mitigation
    - will Conduct side-by-side benchmarking to compare efficiency, inference speed, and segmentation quality.
    - Optimize Transformer models via task-specific fine-tuning on segmentation datasets.
    - If results are marginal, prioritize LSTM optimizations over Transformer deployment.

Dataset Expansion Might Introduce Bias

- Risk: Expanding datasets for Cantonese, Mon, Shan, and other DDLs may lead to imbalanced training and unintended biases.
- Mitigation
    - will Use diverse, representative datasets covering multiple dialects and contexts.
    - Apply data augmentation techniques to improve generalization.
    - Perform cross-validation on unseen datasets to detect and mitigate biases.

ICU4X Integration Might Not Be Feasible Within GSoC Timeline

- Risk: ICU4X's Rust-based integration may require significant pipeline modifications, making it difficult to complete within GSoC.
- Mitigation:
    - Treat ICU4X as a stretch goal, focusing on model improvements first.
    - Document the integration process so it can be continued post-GSoC if necessary.
    - Collaborate with Unicode maintainers early to identify feasibility and blockers.

Testing

Develop a Test Suite for evaluating segmentation accuracy, speed, and memory efficiency.
Benchmarking Compare Dictionary, LSTM, and AdaBoost models against newly trained models.
Cross-validation Ensure model generalization across different languages.
Performance Metrics-

- **Accuracy -**How well the model segments words correctly.
- **Size-**Storage and memory efficiency.
- **Inference Speed-** Latency for real-time segmentation.

Reference Projects

These are the list of projects which we can refer to while implementation.
These projects provide insights into AI-based word segmentation, text tokenization, and Unicode-based language processing.

1. ICU4X Word Segmentation
   Repo:- https://github.com/unicode-org/icu4x
   Relevance: ICU4X is Unicode's internationalization library for Rust, and integrating an optimized segmentation model with ICU4X is a stretch goal of this project.
2. MeCab (Japanese Morphological Analyzer)
   Repo: https://github.com/taku910/mecab
   Relevance: MeCab is widely used for Japanese word segmentation and morphological analysis. Exploring its techniques can provide useful insights for improving segmentation accuracy.
3. pyThaiNLP (Thai Language Processing)
   Repo: https://github.com/PyThaiNLP/pythainlp
   Relevance: A specialized library for Thai NLP tasks, including word segmentation. This project can serve as a baseline for evaluating Unicode models on Thai text.
4. WordPiece & SentencePiece Tokenizers
   Repo (SentencePiece): https://github.com/google/sentencepiece
   Relevance: Used in transformer-based models (like BERT) for subword tokenization. SentencePiece's unsupervised learning approach could inspire enhancements to Unicode's segmentation models.
5. LSTM-Based Word Segmentation (TensorFlow/NLP)
   Repo: https://github.com/diasks2/lstm-seq2seq-word-segment
   Relevance: Demonstrates LSTM-based sequence-to-sequence segmentation, which is similar to the approach used in Unicode's LSTM word segmentation model.

Timeline

My university final exams conclude by the end of April, and I don't have any other commitments during the summer, so I'll be available to work on Unicode full-time if selected. I can dedicate 7-8 hours per day, totaling over 40 hours per week. Since I've already raised a PR and analyzed existing models and have project setup it won't require much time to understand project and start working on implementation proposed by me . Additionally, I have a strong background Python and it's frameworks, and I'm familiar with TensorFlow and Being part of the AI club as a core member, I have practical experience in fine tuning

and model training . I'll use my free time in April to enhance my proficiency in model training and fine tuning further.
I am planning to complete this project in four phases.
Here is the detailed timeline-

Phase 1 (May 1 – May 20): Model Benchmarking & Test Suite Development

---

- Get in touch with mentors (**Younies, Shane, Sahand**) and discuss **design decisions**.
- Finalize the evaluation metrics (**accuracy, speed, model size**) for benchmarking.
- Develop a **test suite** to systematically evaluate different segmentation models.
- Run benchmark tests on **existing Unicode models** (Dictionary, LSTM, AdaBoost) on multiple languages (Japanese, Thai, Cantonese).
- Document findings and initial performance comparisons.

Buffer: 3-5 days to refine test cases, analyze results, and incorporate mentor feedback.

Phase 2 (May 20 – June 5): Model Optimization & Training

- Use insights from benchmarking to **fine-tune hyperparameters** and improve model performance.
- Experiment with **quantization, pruning, knowledge distillation** to optimize models.
- If required, train models on **additional data sources** for improved accuracy.
- Test refined models on **dummy datasets** to ensure stability.
- Present findings to **mentors and community** for feedback on model architecture and optimizations.

Buffer: 4-5 days to account for debugging, unexpected training delays, or data inconsistencies.

Phase 3 (June 5 – July 5): Developing & Implementing an Improved Segmentation Model

---

- Implement **a new AI-based segmentation model** aimed at outperforming current Unicode models.
- Design a **clean and modular interface** for AI-based segmentation.
- Perform **real-world testing** on complex scripts, especially **digitally disadvantaged languages (DDLs)**.
- Conduct **performance testing** to measure model inference time and efficiency.
- Improve the model based on community feedback.
- Begin writing **technical documentation** for code, dataset handling, and evaluation processes.

Buffer: 5-7 days for performance testing, code refactoring, or additional debugging.

Phase 4 (July 5 – July 30): Final Optimization & Deployment

- Implement **final optimizations** based on previous testing phases.
- Perform **rigorous testing** across various datasets.
- Fix any **remaining issues or model inconsistencies**.
- Integrate the optimized model into **ICU4X (Rust)** (if feasible).
- Set up **automated tests and documentation** for reproducibility.
- Submit the **final report**, code, and model performance results.

Buffer: 6-8 days for last-minute bug fixes, documentation improvements, or feedback incorporation.

Final Buffer (July 30 – August 25)

- Address any **unforeseen technical challenges** or performance issues.
- Additional **fine-tuning** based on mentor/community feedback.
- Ensure **full documentation and usability improvements** before the final submission.

Final Buffer: 10-15 days to accommodate last-minute feedback and ensure the best possible submission.

I have kept 2-3 weeks of buffer time if things go south or maybe Implementing specific portion takes a lot of time.

About me

- Name - Anushka
- University – VIT Bhopal University
- Email – anushkachaudhary19128@gmail.com
- GitHub Username: anushka-cseatmnc
- Time Zone - IST (GMT +5:30)
- Preferred Language For Communication: English
- Linkedin - https://www.linkedin.com/in/anushka-chaudhary-73416127a/

I am **Anushka Chaudhary**, an AIML enthusiast currently 2nd year student pursuing integrated M.Tech AI at VIT Bhopal university . Passionate about NLP and inclusive language technologies. I have experience in-

- **AI/ML Model Development** (Python, TensorFlow, PyTorch)
- **Natural Language Processing** (sequence models, segmentation)
- **Research & Optimization** (model fine-tuning, efficiency improvements)
- **Open Source Contributions** (Unicode segmentation, TensorFlow-free weight conversions)

My work aims to bridge the gap between AI and linguistics, ensuring **underrepresented languages** receive the same digital accessibility as widely used languages.

Pull Request

I have already explored the **existing models** and contributed to the **Unicode segmentation repository**.
PR:- **Added TensorFlow-free npy and h5 weight conversions** #36
https://github.com/unicode-org/lstm_word_segmentation/pull/36
Improved model weight handling to **reduce dependency on TensorFlow**. (Still in progress , need some refinements.)
Enhanced model **compatibility and efficiency** in Unicode's segmentation pipeline.
Demonstrated **hands-on experience with Unicode's codebase**,for making future contributions smoother.

I'm truly passionate about contributing to open source projects. I'm particularly intrigued by Unicode because it's used by the vast majority of internet users. Contributing to Unicode would be an exciting experience because it impacts billions of people worldwide. Currently, I'm exploring open source , and I've found the Unicode/icu4x repository fascinating. I plan to contribute to it in the near future. Additionally, I'll always be available for any future changes or extensions to this project.

Past Experience

I have previously worked with Vision Transformers (ViTs) in my project 'Vision-Transformer-and-Dense-Model-on-CIFAR-100,' where I analyzed and compared the performance of Transformers against traditional dense models for image classification. This experience has given me a strong understanding of attention mechanisms, model optimization, and deep learning frameworks like PyTorch and TensorFlow. I am excited to apply these skills to NLP-based segmentation tasks in this GSoC project, particularly when exploring Transformer-based alternatives to LSTMs.

Repo link :- https://github.com/anushka-cseatmnc/Vision-Transformer-and-Dense-Model-on-CIFAR-100

Availability

I am available full-time during the GSoC period and will dedicate **40+ hours per week** to this project. I will ensure timely progress, regular mentor communication, and well-documented contributions.

Future Scope & Post-GSoC Plans

I will document improvements and onboard new contributors.

Project Continuation

- Expand support for more underrepresented languages.
- Integrate the improved model into real-world applications like chatbots, OCR, and text-to-speech systems.
- Maintain and update the test suite with new datasets and evaluation metrics.

Involvement in the Unicode Community

- Continue contributing to ICU4X by integrating the optimized model.
- Assist new contributors through code reviews and discussions.
- Stay active in Unicode's NLP projects and related research.

Next Steps After GSoC

- Explore transformer-based models like BERT and ALBERT for segmentation.
- Improve handling of languages with mixed scripts.
- Optimize the model for mobile and low-power devices.

References

---

1. https://docs.google.com/document/d/e/2PACX-1vQwQhjcjYENWYTfA4_FEsyVPZzqhp-QAyeVfPcYR9x0RFqL4mhby1hWKatjiG6Itwkuz8Q4ZXSF_IZz/pub

2. https://docs.google.com/document/u/2/d/e/2PACX-1vQbj0-VFkRjYdnivuPPXuHM3IW4LuHxK6E0LVO3O8ZU_-k8CYH_eFMZ_IwFg_r-oBw3FCEOmHCb5jrn/pub

3. https://developers.google.com/open-source/gsoc/timeline

4. https://google.github.io/gsocguides/student/writing-a-proposal

5. https://github.com/unicode-org/icu4x/discussions/4727#discussioncomment-8955865