# KeemenaPreprocessing.jl — Unicode-Robust Cleaning, Multi-Level Tokenisation & Streaming Offset Bundling for Julia NLP

July 7 2025

## Summary

KeemenaPreprocessing.jl begins where raw text first enters a research workflow, applying a carefully chosen set of cleaning operations that work well for most corpora yet remain fully customisable. By default the toolkit lower-cases characters, folds accents, removes control glyphs, normalises whitespace, and replaces URLs, e-mails, and numbers by sentinel tokens; each rule may be toggled individually through an optional `PreprocessConfiguration`, so users can disable lower-casing for case-sensitive tasks or preserve digits for OCR evaluation without rewriting the pipeline.

After cleaning, the same configuration drives tokenisation. Keemena ships byte-, character-, and word-level tokenisers and will seamlessly wrap a user-supplied function—allowing, for instance, a spaCy segmentation pass when language-specific heuristics are required (Honnibal et al. 2020). Multiple tokenisers can operate in one sweep, so a single corpus pass can yield both sub-word pieces for a language model and whitespace tokens for classical bag-of-words features. Each token stream is accompanied by dense offset vectors: words are anchored to their byte and character positions, sentences and paragraphs are delimited explicitly, and a cross-alignment table keeps byte <-> char <-> word mappings exact. This design guarantees that every higher-level span can be traced unambiguously back to the source bytes, a property indispensable for annotation projection and reversible data augmentation.

All artefacts—clean strings, token-ids, offset vectors, vocabulary statistics, and alignment tables are consolidated into a single `PreprocessBundle`. The bundle can be saved or loaded with one function call using the JLD2 format, making it a drop-in dependency for downstream embedding or language-model pipelines inspired by word2vec (Mikolov 2013). For modest datasets, the entire pipeline executes in a single statement; for web-scale corpora, KeemenaPreprocessing's streaming mode processes fixed-size token chunks in constant memory while

still accumulating global frequency tables. Thus, whether invoked with default settings for a quick experiment or finely tuned for production, KeemenaPreprocessing.jl offers a cohesive, Julia-native path from raw text to analysis-ready data (Bezanson et al. 2017). Many of these principles are introduced in (Bird, Klein, and Loper 2009);

## Statement of Need

Natural-language ML pipelines depend on reliable, reproducible preprocessing. Popular toolkits such as spaCy (Honnibal et al. 2020), Stanford CoreNLP (Manning et al. 2014), and Gensim (Rehrek and Sojka 2010) are Python or Java-centric, require heavyweight installations, and assume the full corpus fits in memory or on a local filesystem. While WordTokenizers.jl provides basic tokenisation for Julia (Kaushal et al. 2020), Julia users still lack an integrated, streaming pipeline that:

- Scales beyond RAM through chunked streaming.
- Tracks fine-grained offsets so models can mix sub-word and sentence-level features.
- Lives entirely in Julia, avoiding Python/Java dependencies and enabling zero-copy interop with Julia's numerical stack.

KeemenaPreprocessing.jl fills this gap, letting researchers preprocess billions of tokens on commodity hardware while retaining compatibility with embedding or language-model training workflows inspired by word2vec (Mikolov 2013).

## Acknowledgements

Thanks to the Julia community for their continued support of open-source scientific computing.

## References

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2017. "Julia: A Fresh Approach to Numerical Computing." *SIAM Review* 59 (1): 65–98. https://doi.org/10.1137/141000671.

Bird, Steven, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* " O'Reilly Media, Inc.".

Honnibal, Matthew, Ines Montani, Sofie Van Landeghem, Adriane Boyd, and others. 2020. "SpaCy: Industrial-Strength Natural Language Processing in Python."

Kaushal, Ayush, Lyndon White, Mike Innes, and Rohit Kumar. 2020. "Word-Tokenizers. Jl: Basic Tools for Tokenizing Natural Language in Julia." *Journal of Open Source Software* 5 (46): 1956.

Manning, Christopher D, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. "The Stanford Corenlp Natural Language Processing Toolkit." In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60.

Mikolov, Tomas. 2013. "Efficient Estimation of Word Representations in Vector Space." *arXiv Preprint arXiv:1301.3781* 3781.

Rehrek, Radim, and Petr Sojka. 2010. "Software Framework for Topic Modelling with Large Corpora."