BenchmarkDataNLP.jl: Synthetic Data Generation for NLP Benchmarking

25 January 2025

Summary

BenchmarkDataNLP.jl is a package written in Julia Lang for generating synthetic text corpora that can be used to systematically benchmark and evaluate Natural Language Processing (NLP) models such as RNNs, LSTMs, and Large Language Models (LLMs). By enabling users to control core linguistic parameters such as the alphabet size, vocabulary size, grammatical expansion complexity, and semantic structures this library can help users test (as well as evaluate or debug) NLP models. Instead of exposing many of the key internal parameter choices to the user a minimal number are so that users do not have to focus on how to correctly configure the generation process. The key value for the user is the *complexity* integer value which controls the size of the alphabet, vocabulary and grammar expansions. This allows a range starting from 1 so that very simple text corpora can be generated. For example at complexity=1 there are 5 letters in the alphabet and 10 words used in the vocabulary with 2 grammar roles when a Context Free Grammar generator is select. Users can choose how many independent productions are desired which are each supplied as entries in a .jsonl file. The defaults provided in the documentation should suffice for most use cases.

The generator methodology approaches are:

- Context Free Grammar (Vugt 1996)
- Resource Description Framework (RDF, triple store) (Faye, Cure, and Blin 2012)
- Finite State Machine (Maletti 2017)
- Template Strings (Copestake 1996)

Each method offers different options to the user. With the Finite State Machine approach, the function *generate_fsm_corpus* allows for users to produce a deterministic set of productions so that upon successful training accuracies of 100% can be achieved. An application of this is that different models can be tested on the amount of computational requirements they have for a particular

value of complexity. The 4 approaches cover a wide range of text expansion production methods that does not have parameter estimations as part of their usage requirements. The letters of the alphabet begin in the *HANGUL* start (44032, 0xAC00) and does not apply repetition of labels with integer identifiers at the end which are only convenient for implementations but may not be ideal for some tokenization approaches.

This package also hopes to bring down the cost required when exploring different models and architectures that would require large amounts of energy, time and finances to train on large natural language datasets (Samsi et al. 2023). Risks of this nature can often be expected when very novel approaches are explored.

Statement of need

Performance evaluation of NLP systems often hinges on realistic datasets that capture the target domain's linguistic nuances. However, training on large-scale, text corpora can be expensive or impractical, especially when testing very particular aspects of language complexity or model robustness. Synthetic data generation helps bridge these gaps by:

1. **Reproducibility**: Controlled parameters (e.g., sentence length, grammar depth, or concept re-use) allow reproducible experiments.
2. **Customization**: Researchers can stress-test models by systematically varying language properties, such as the number of roles in a grammar or the frequency of filler tokens.
3. **Scalability**: Large-scale data can be generated for benchmarking advanced architectures without the need for extensive, real-world data collection.
4. **Targeted Evaluation**: By manipulating semantic structures (for example, adding context continuity with RDF triples or specialized placeholders), researchers can investigate whether models capture specific linguistic or contextual features.

Although several libraries and benchmarks (e.g., GLUE, SuperGLUE, and others) provide curated datasets, **BenchmarkDataNLP.jl** offers a unique approach by allowing *fine-grained control* of the underlying complexity of a synthetic corpus generation process. This capability is especially valuable when exploring model failure modes or for rapid prototyping of new model architectures that require specialized text patterns. It is hoped that this will bring down the cost for initial prototyping of new model architectures and allow a greater exploration. This can also help compare different modeling approaches.

Acknowledgements

We thank the Julia community for their continued support of open-source scientific computing.

References

Copestake, Ann. 1996. "Applying Natural Language Processing Techniques to Speech Prostheses." In *Working Notes of the 1996 AAAI Fall Symposium on Developing Assistive Technology for People with Disabilities*.

Faye, David C, Olivier Cure, and Guillaume Blin. 2012. "A Survey of RDF Storage Approaches." *Revue Africaine de Recherche En Informatique Et Mathématiques Appliquées* 15. https://doi.org/10.46298/arima.1956.

Maletti, Andreas. 2017. "Survey: Finite-State Technology in Natural Language Processing." *Theoretical Computer Science* 679: 2–17. https://doi.org/https://doi.org/10.1016/j.tcs.2016.05.030.

Samsi, Siddharth, 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." In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–9. IEEE. https://doi.org/10.1109/HPEC58863.2023.10363447.

Vugt, Nikè van. 1996. "Generalized Context-Free Grammars." *Internal Report*.