xLM

A Unified Framework for Non-Autoregressive Language Models

Dhruvesh Patel **Durga Prasad Maram** **Sai Sreenivas Chintha**
Benjamin Rozonoyer **Andrew McCallum**

UMassAmherst
Manning College of Information & Computer Sciences

Motivation

There is a lot of interest in non-autoregressive text generation. But each implementation has its own bespoke boilerplate code for data pipeline, training, and evaluation—making it difficult to perform systematic comparisons. xLM aims to provide a shared, modular foundation to help researchers focus on the model, not the scaffolding.

Autoregressive LM

One token appended left to right per step.

```
The quick …
The quick brown …
The quick brown fox …
The quick brown fox jumps
```

Masked Diffusion LM

Positions unmasked in parallel over steps.

```
The _ _ _ _ _ _
The _ _ brown _ _ _
The quick brown _ _ jumps
The quick brown fox jumps
```

Insertion LM

New tokens inserted at chosen positions per step.

```
The fox
The brown fox
The brown fox jumps
The quick brown fox jumps
```

Design Principles

Maximal Independence

Each model lives in its own self-contained package—usable entirely outside xLM, no refactoring required.

Composition over Inheritance

The Harness holds typed slots for Model, Loss, Predictor, and Collator and simply delegates—swap any one without touching the others.

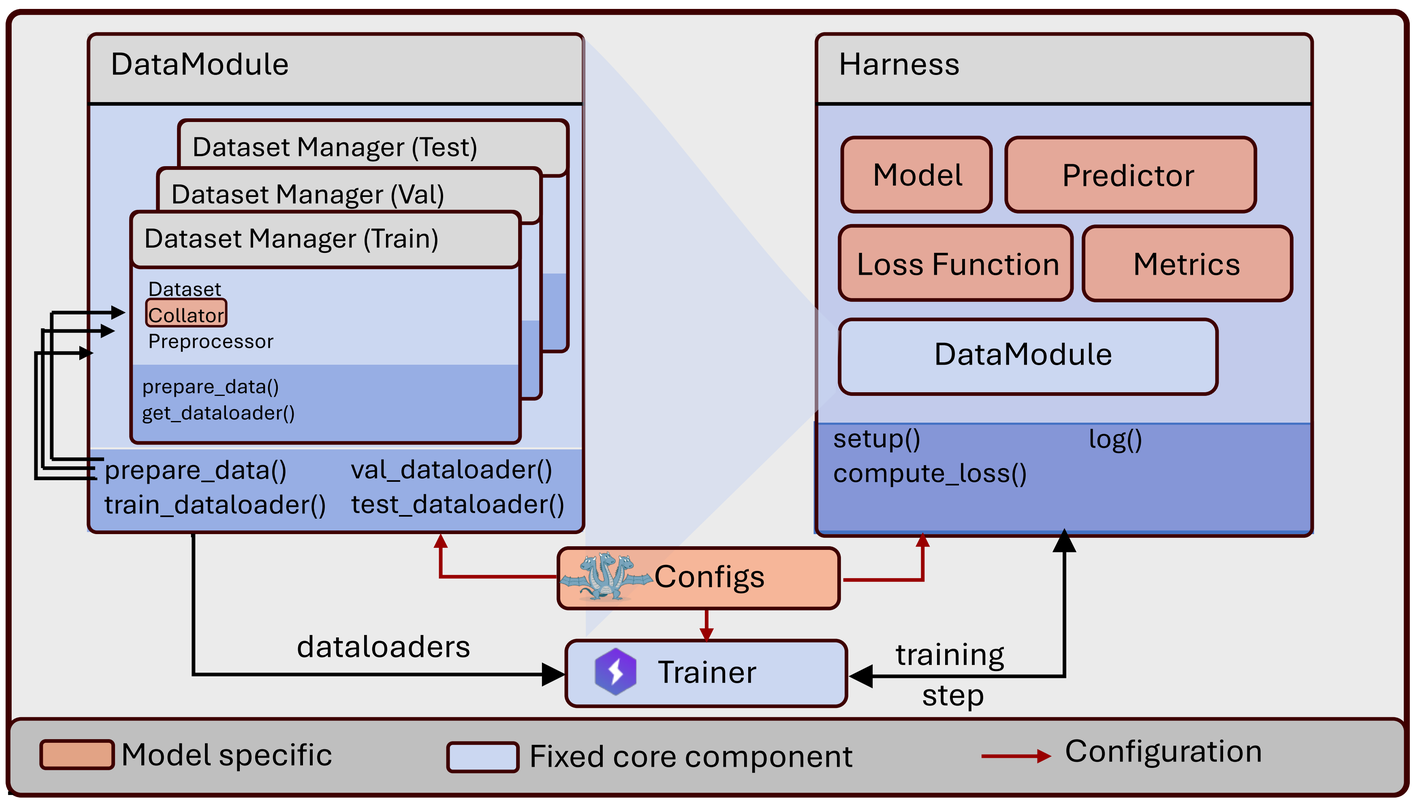
Copy over Branching

Explicit copies reduce complexity, keep each model self-documenting, and enable LLM-assisted rapid prototyping.

Arbitrary Code Injection

Hydra config slots accept *any* class at runtime—zero Python changes to replace an entire component.

Architecture



Core components (Harness, DataModule) are model-agnostic. **Model-specific slots** (Model, Loss, Predictor, Collator) are injected via Hydra config—full component swapping without a single line of Python change.

Directory Structure

Example directory structure for a standalone model.

```
ilm/ ..................................................... model root
├── __init__.py
├── types_ilm.py ..................................................types
├── model_ilm.py ..................................... network
├── loss_ilm.py ..................................... loss
├── predictor_ilm.py ..................................... inference
├── datamodule_ilm.py .....................................collators
├── metrics_ilm.py ..................................... metrics
└── configs/ ..................................... hydra configs
    ├── model/
    │   └── ilm.yaml
    ├── model_type/
    │   └── ilm.yaml
    ├── collator/
    │   └── default_ilm.yaml
    ├── datamodule/
    │   └── star_easy_ilm.yaml
    └── experiment/
        └── star_easy_ilm.yaml
├── setup.py
└── README.md
```

`xlm-scaffold ilm`

Workflows

The main job types.

```
xlm job_type=<JOB> job_name=<NAME> experiment=<CONFIG>
```

job_type options:

- `prepare_data` — download & cache datasets
- `train` — training loop
- `eval` — evaluation
- `generate` — inference
- `push_to_hub` — upload to HF Hub

Add `debug=overfit` for quick single-batch debug.

At a Glance

Packages

```
pip install xlm-core
pip install xlm-models
```

Code

Links dhruveshp.com/xlm-core/latest/

github.com/dhruvdcoder/xlm-core

Get Started

Install

```
pip install xlm-core xlm-models
```

Train / Eval / Generate

```
xlm job_type=train \
  job_name=<NAME> \
  experiment=<CONFIG>
```

Scaffold a new model

```
xlm-scaffold my_model
```

Push to HuggingFace Hub

```
xlm job_type=push_to_hub \
  +hub.repo_id=<REPO>
```

Key Features

- **Lightning-powered** — DDP, AMP, logging out of the box
- **Hydra configs** — swap entire components from YAML, no code changes needed
- **Scaffolding** — `xlm-scaffold` generates a full typed model skeleton
- **HF Hub** — push trained weights in one command
- **Common modules** — RotaryTransformer, DiT, noise schedulers under `xlm.modules`
- **Useful callbacks** — EMA, checkpointing with thinning, crash recovery

Available Models

`mlm`	Masked LM	BERT-style unmasking
`ilm`	Insertion LM	Iterative token infilling
`arlm`	Autoregressive LM	Left-to-right generation
`mdlm`	Masked Diffusion LM	Parallel unmasking
`flexmdm`	Flexible Masked Diffusion LM	Variable-length unmasking

More models coming soon! Contributions welcome!

Preconfigured Datasets

`lm1b`	Billion-word benchmark
`openwebtext`	OpenWebText
`star`	Star graphs
`SAFE Molecules`	SAFE strings for small molecules
`sudoku-extreme`	Large Sudoku dataset with hard puzzles and solver trajectories

Contributions welcome!

References

[1] William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019.

[2] Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Gopal Paliwal, Weili Nie, and Arash Vahdat. GenMol: A Drug Discovery Generalist with Discrete Diffusion. In *Forty-Second International Conference on Machine Learning*, June 2025.

[3] Dhruvesh Patel, Aishwarya Sahoo, Avinash Amballa, Tahira Naseem, Tim GJ Rudner, and Andrew McCallum. Insertion language models: Sequence generation with arbitrary-position insertions. *arXiv preprint arXiv:2505.05755*, 2025.

[4] Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M. Rush, Yair Schiff, Justin T. Chiu, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

[5] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.