PLAQUETTE · DISPONIBLE · VERSION 0.2.1

lm-resizer

Vos agents paient plein tarif pour lire des journaux de test. lm-resizer garde le signal et laisse tomber le bruit, avant que le modèle ne le voie.

Un agent de développement dépense une part énorme de sa fenêtre de contexte en sorties brutes : tests, journaux, différences, listings, JSON. lm-resizer s'intercale — en enveloppe de commande, en serveur MCP ou en mandataire HTTP — filtre selon la commande exécutée, compresse en gardant ce qui répond à la question posée, et conserve la sortie complète récupérable par un simple lien.

État : Version 0.2.1 publiée le 9 septembre 2026 sous licence Apache 2.0, paquet npm @phuetz/lm-resizer. Chaîne complète vérifiée le 23 juin 2026 devant Mistral, Ollama, DeepSeek, OpenRouter et xAI.

À qui ça sert

Les équipes qui paient leurs tokens

Chaque exécution de tests renvoyée intégralement au modèle est une facture et une fenêtre de contexte gaspillées.

Ceux qui travaillent sur un modèle local

Sur une petite fenêtre de contexte, la sortie brute d'un lanceur de tests suffit à saturer la session. Le filtre rend le modèle local utilisable.

Les intégrateurs d'agents

Enveloppe de commande, outils MCP, API HTTP ou mandataire compatible OpenAI et Anthropic : quatre points d'entrée pour le même moteur.

Ce que ça change concrètement

1. **Le bruit disparaît, les échecs restent**

   Les filtres connaissent git, cargo, ripgrep, vitest, jest et d'autres familles de commandes : les lignes d'échec, les chemins de fichiers et les résumés sont préservés, les milliers de lignes de progression s'effacent.

2. **La preuve complète reste à un lien**

   La sortie entière est stockée localement, dans une base SQLite, et récupérable par son empreinte. Compresser n'est pas perdre : rien n'est détruit, tout est retrouvable.

3 **Une compression qui connaît la question**

Quand il faut couper, ce qui répond à la demande en cours est gardé en priorité, au lieu d'une troncature aveugle au milieu d'une trace d'erreur.

4 **Quatre manières de l'installer**

Enveloppe de commande, serveur MCP pour Claude Code et Codex, API HTTP, ou mandataire transparent devant OpenAI, Anthropic, Bedrock et Vertex.

5 **Un mode qui n'exécute jamais rien**

Le mode sortie d'outil accepte un flux déjà produit par l'hôte, sans lancer la moindre commande : l'agent garde son bac à sable, lm-resizer ne fait que réduire.

Comment ça marche

Le moteur est natif Rust et reste sur votre machine. Quatre temps, entre la commande et le modèle.

1

Sortie brute

Tests, journaux, différences, résultats de recherche, sortie de gestionnaire de paquets, gros JSON, trafic du fournisseur.

2

Détection et filtrage

Le type de contenu est reconnu, puis un filtre propre à la famille de commande retire la répétition et garde les échecs.

3

Compression et délestage

Minification, résumés, rétention guidée par la question ; les gros volumes sont déportés dans une base SQLite locale avec une empreinte de récupération.

4

Contexte utile

Le modèle reçoit les erreurs, les chemins, les résumés et les liens de récupération. La preuve entière reste accessible par la commande retrieve.

Aucune donnée ne sort de la machine : la base de récupération est un fichier local, et la classification par apprentissage automatique est facultative et désactivée par défaut.

Chiffres vérifiables

Chaque chiffre ci-dessous est daté et provient du dépôt ou d'une mesure publiée. Rien n'est extrapolé.

222 247 tokens

économisés sur une seule exécution de cargo test, soit 398 commandes et 1,23 Mo ramenés à 372 Ko

23 juin 2026

chaîne vérifiée de bout en bout devant Mistral, Ollama en local, DeepSeek, OpenRouter et xAI

Source : README du dépôt, mesure sur une exécution complète.

Source : README du dépôt, section de vérification datée.

0.2.1

version du paquet npm @phuetz/lm-resizer, publiée le 9 septembre 2026

Source : Registre npm public, champ de date de publication.

Pour démarrer

Le moteur se compile avec Rust 1.80 ou plus récent. Le module WebAssembly, lui, s'installe depuis npm, et le greffon Claude Code en deux commandes.

1. Compiler l'outil

```
git clone https://github.com/phuetz/lm-resizer.git
cd lm-resizer
cargo build --release
```

2. Envelopper une commande bruyante

La commande s'exécute normalement, sa sortie arrive filtrée.

```
lm-resizer exec -- cargo test
lm-resizer exec -- npm test
lm-resizer retrieve <empreinte-ccr>   # la sortie complète, intacte
```

3. Le brancher sur votre agent

```
lm-resizer install --client claude --scope project
lm-resizer install --client codex  --scope global

claude plugin marketplace add phuetz/lm-resizer
claude plugin install lm-resizer@phuetz-tools
```

4. Ou le mettre devant le fournisseur

Mandataire compatible OpenAI et Anthropic, sur votre boucle locale.

```
lm-resizer serve --bind 127.0.0.1:8787

npm install @phuetz/lm-resizer   # le même moteur en WebAssembly
```

Ce qui n'est pas encore prêt

Nous préférons l'écrire noir sur blanc plutôt que de le faire découvrir après la signature.

— Les filtres livrés couvrent les familles de commandes les plus coûteuses ; un filtre propre à un écosystème maison reste à écrire, à partir des modèles fournis.

— La classification par apprentissage automatique (Magika, ONNX) est une option de compilation, désactivée par défaut.

— L'outil n'embarque ni collecteur de télémétrie ni tableau de bord : les mesures restent locales.

Licence, dépôt et contact

Licence
Apache 2.0, sans clause additionnelle. Le paquet npm @phuetz/lm-resizer suit la même licence.

Où le trouver

- github.com/phuetz/lm-resizer (https://github.com/phuetz/lm-resizer)
- npm install @phuetz/lm-resizer (https://www.npmjs.com/package/@phuetz/lm-resizer)
- Documentation du projet (https://phuetz.github.io/lm-resizer/)
- → Fiche produit sur ce site

Éditeur
AGILE UP — Structure d'ingénierie logicielle indépendante, Vincennes.

Contact
contact@agile-up.com
agile-up.com