Reproducibilité des résultats de recherche à l'aide de GNU/Guix

Application à la gestion de bibliothèques matérielles

Cayetano Santos

15èmes Journées Informatiques IN2P3/IRFU

23 Septembre 2024

contexte

"Plus de 70% des chercheurs ont essayé de reproduire les expériences d'un autre scientifique, sans y parvenir."

"Plus de la moitié n'ont pas réussi à reproduire leurs propres expériences."[1]

intelligence ? . . .

Des chercheurs ont testé GPT-4 sur 40 tâches de «fausses croyances» qui ont été utilisées pour évaluer les capacités de théorie de l'esprit chez les enfants et ont découvert que le GPT-4 résolvait presque toutes ces tâches.

L'auteur a pris ces résultats comme un support pour affirmer :

"Les grands modèles linguistiques tels que GPT-4 ont acquis une «théorie de l'esprit» – une capacité à comprendre les croyances et les motivations des gens."[2]

. . . pas vraiment

Une étude de suivi [3] a effectué les mêmes tests et effectué le type d'expériences systématiques et soigneusement contrôlées.

Ils ont découvert que plutôt que d'avoir de solides capacités de théorie de l'esprit, GPT-4 et d'autres modèles de langage semblent plutôt s'appuyer sur des « heuristiques superficielles » pour effectuer les tâches de l'article original.

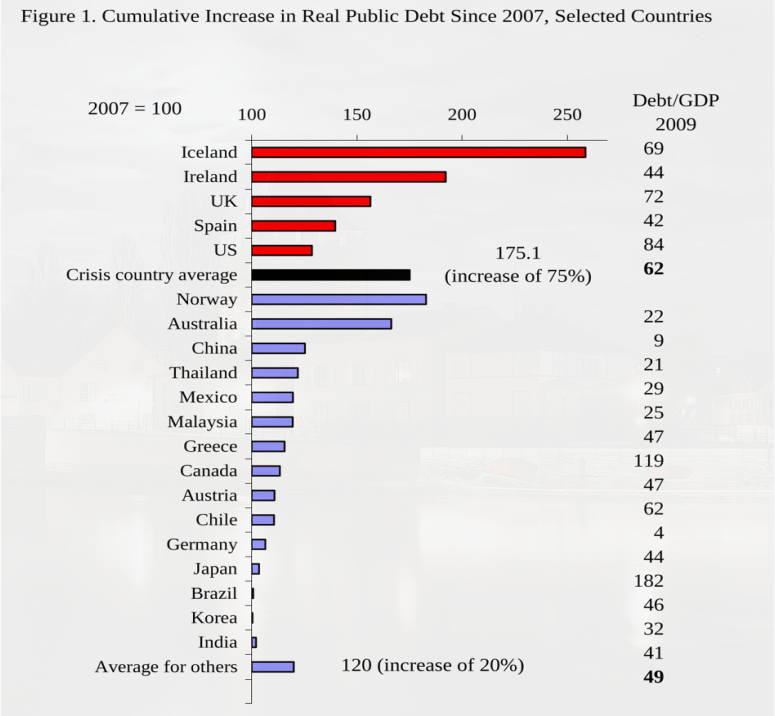
> *«Nous mettons en garde contre le fait de tirer des conclusions à partir d'exemples anecdotiques, de tester sur quelques points de référence et d'utiliser des tests psychologiques conçus pour les humains pour tester les modèles [d'IA]»*[4]

décroissance non ouverte

Il s'avère que cette conclusion repose en partie sur une erreur d'Excel. [1]

L'analyse avait exclu [2] une poignée de données critiques [3].

"Les pays qui se sont endettés à hauteur de plus de 90 % de leur produit intérieur brut, ont subi une forte baisse de leur croissance économique"[5]



1. Influential Reinhart-Rogoff economics paper suffers spreadsheet error
2. The Reinhart-Rogoff error – or how not to Excel at economics
3. The Excel formula error that initiated austerity policies after the crisis

science ouverte

Plusieurs types de reproductibilité dans les résultats de recherche [6]

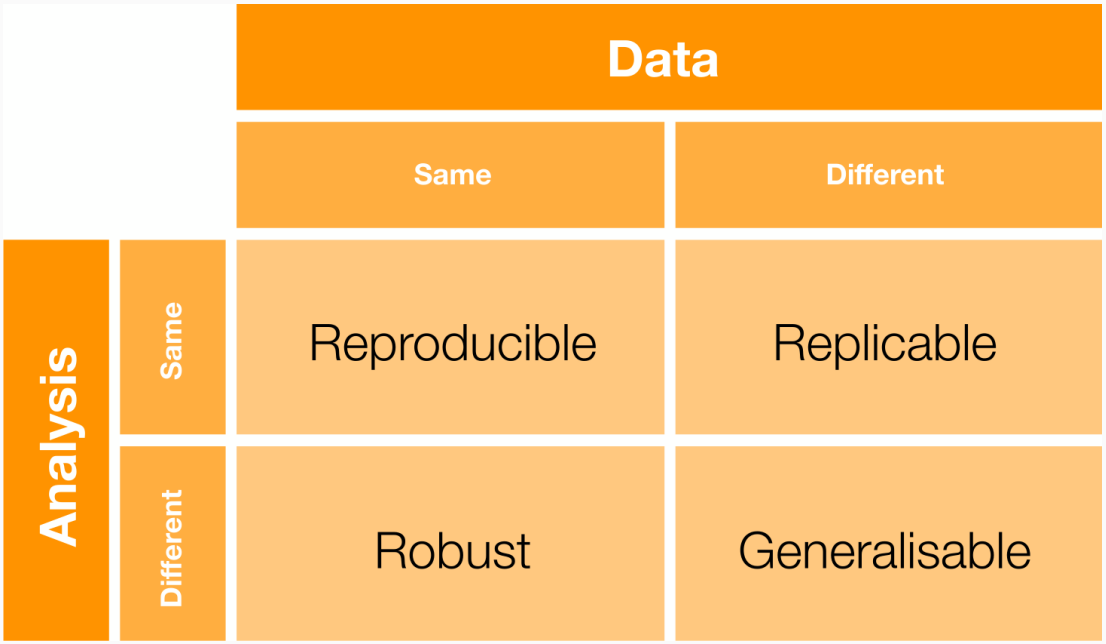
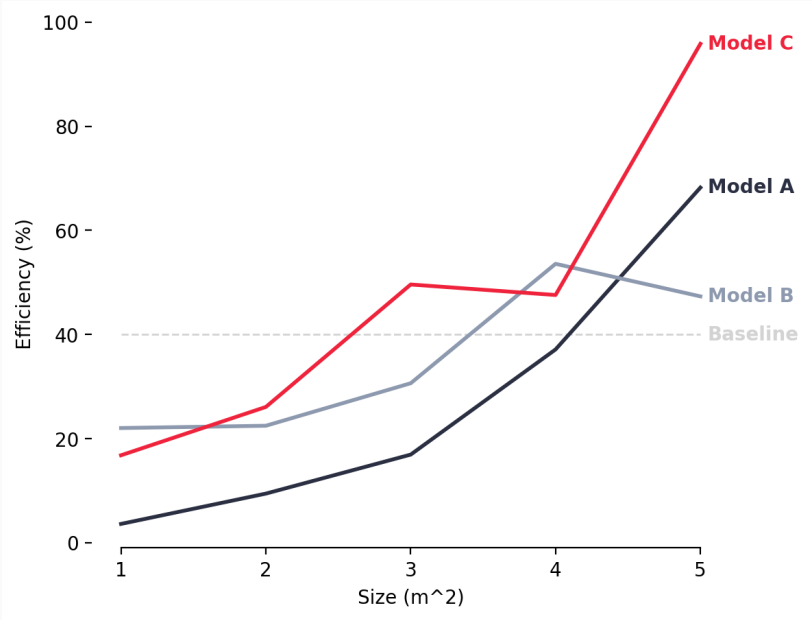


Image Scriberia for The Turing Way community, under a CC-BY licence. [7]

question de confiance ?

On va chercher à répondre à la question . . . [4]



- vous y croyez ?
- vous me faites confiance ?
- je me suis trompé ?
- j'arrive à reproduire mon plot ?
- dans un an ?
- quelqu'un d'autre y arrive ?
- et sur un OS différent ?

4. Image: Making Plots Pretty

Partager son code ? . . . pas suffisant [8]

- sous une licence libre (*quels sont mes droits ?*)
- contrôle de versions (*quelle version ?*)
- y avoir accès sur le long terme avec SWH (*où ?*) [9]

ancillaires

- identifier sans ambiguïté les données associées (*quels binaires ?*)
- documentation minimal (*comment ?*)
- reconnaissance des sources externes (*d'où ça sort ?*)
- références bibliographiques `.bib` (*pourquoi ?*)

. . . et surtout, fournir un contexte

- environnent (*où ça tourne ?*) -> GNU/GUIX !!

GNU/Guix

Gestion logicielle

```
guix search python
guix install/remove emacs/vim
```

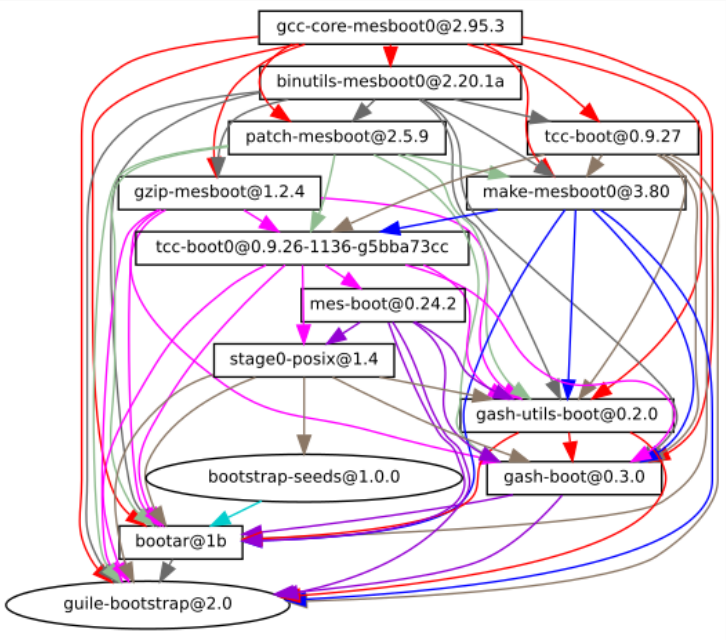
- externe sur n'importe quelle (*systemd*) distri linux
- cohabitation sans conflit avec gestionnaire local
- *store* central de paquets + liens symboliques
- accès à des logiciels non disponibles en local
- intégration avec Software Heritage
- scheme comme langage d'extension

déterminisme

Toute la distri est compilée à partir d'un noyau binaire de confiance. [5]

-> approche fonctionnelle de la gestion de paquets <-

paquet : `code source` + `recette` + `dépendances`



Reproductible jusqu'au bout (*full source bootstrap*)

Noyau de 357-bytes -> 22k paquets

Binaires toujours identiques

5. The Full-Source Bootstrap: Building from source all the way down

channels.scm

Guix est un repo git, `guix pull` pour mettre à jour

- code
- descriptif des paquets

On va fixer la `version` (commit) du canal (*quel guix ?*)

```
guix describe --format=channels -p $GUIX_PROFILE > $TMPDIR/channels.scm
# guix pull -C $TMPDIR/channels.scm
```

```
(list (channel
        (name 'guix)
        (url "https://git.savannah.gnu.org/git/guix.git")
        (branch "master")
        (commit
          "a5a990d07429cb836c3811930900d6cebdbfb35f")
        ...))
```

Possibilité de rajouter des canaux custom
(repos git avec descriptif de paquets)

manifest.scm

Export de la liste des logiciels installés dans un profile

```
guix package -p $GUIX_PROFILE --export-manifest > $TMPDIR/PROFILE-manifest.scm
```

```
(specifications->manifest
 (list
  "emacs"
  "emacs-org-reveal"
  "emacs-magit"
  "emacs-oauth2"
  ...))
```

Import de la liste des logiciels à installer

```
guix package -p $GUIX_PROFILE -m $TMPDIR/$PROFILE-manifest.scm
```

$GUIX_PROFILE : redéfinition des variables d'environnement

```
export GUIX_PROFILE=$HOME/.guix-profiles/$PROFILE/guix-profile
[ -f "$GUIX_PROFILE/etc/profile" ] && . "$GUIX_PROFILE/etc/profile"
```

Avec un `etc/profile`

```
export PATH=\
"${GUIX_PROFILE:-/gnu/store/svjmjgdbd28rfxljfsh9a7xxm7fqii8l-profile}/bin:${GUIX_PROFILE:-/g
export XDG_DATA_DIRS=\
"${GUIX_PROFILE:-/gnu/store/svjmjgdbd28rfxljfsh9a7xxm7fqii8l-profile}/share${XDG_DATA_DIRS:+
...
```

$P_{python3.1}$

- python 3.1
- gcc 7
- ...

$P_{python3.8}$

- python 3.5
- gcc 9
- ...

$P_{python3.10}$

- python 3.10
- gcc 10
- ...

vraies environnements

Contextes VRAIMENT isolés du reste du système . . .

```
>> which emacs
sh: which: command not found
>> cat
sh: cat: command not found
>> ls
sh: ls: command not found
```

. . . à customiser avec la liste de logiciels souhaité

```
guix shell \
     --container \              # container à la volée
     --network \                # accès réseau
     --emulate-fhs \            # système de fichiers habituel
     --link-profile \           # créer un profil
     -m manifest.scm \          # liste des logiciels à installer
     -- python3 run.py         # commande dans l'environnent
```

time machine

A : on développe et on teste en local dans ***$MYENV***

`Guix` courant appelle une ancienne version (commit) de `guix`

```
guix time-machine \           # guix courant
     -C channels.scm -- \     # ancienne version de guix
     shell \
     --container \
     -m manifest.scm \        # liste de logiciels
     -- COMMANDE
```

Example : reproduire cette exposée à l'identique

```
guix time-machine -C $TMPDIR/channels.scm -- shell --container -m manifest.scm --
↪  emacs ...
```

sur n'importe quel poste et n'importe quand

deploiement

Quand `guix` n'est pas disponible, il est possible de déployer des images `tar.gz`, `docker` et `singularity` déterministes identiques à chaque fois.

```
guix time-machine \
     -C channels.scm -- \
     pack \                # commande pack
     -f FORMAT \
     --save-provenance \   # retour en arrière
     -m manifest.scm
```

B : on teste sur forge gitlab dans le même environnent ***$MYENV***[6]

```
guix time-machine -C channels.scm -- pack -f docker -m manifest.scm
```

```
docker load < $IMAGE
```

```
docker tag $IMAGE:latest gitlab-registry.in2p3.fr/$IMAGE:TAG
```

Finally, pushed to the registry with

```
docker push gitlab-registry.in2p3.fr/$IMAGE:TAG
```

```
# .gitlab-ci.yml
image:
name: gitlab-registry.in2p3.fr/$IMAGE:TAG
```

6. Git Best Practices

C : on deploie sur cluster toujours dans le même environnent ***$MYENV***

```
guix time-machine -C channels.scm -- pack -f singularity -m manifest.scm
```

```
scp $IMAGE cca:~/.
```

Example : déploiement d'un `profil.tar.gz` avec emacs 30 sur un HPC

et bien plus

transactions historique de changements

imports pour la création de paquets automatisée

graph visualisation de dépendances

substituts disponibilité de paquets pre-compilés en ligne

publish partage de substituts

refresh mises à jour

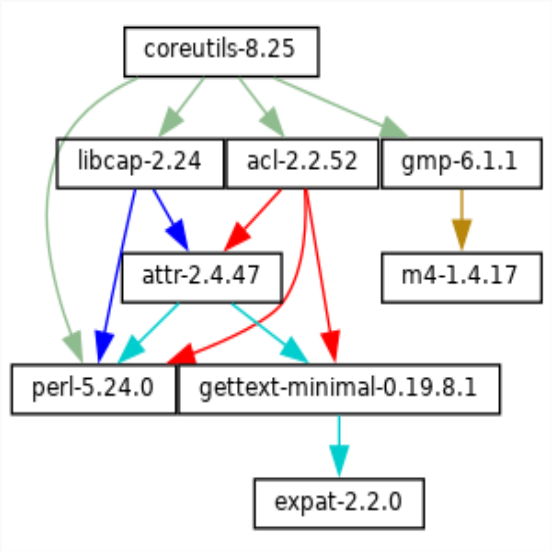
arch support pour des architectures arm, etc.

cas d'usage : gestion de bibliothèques matérielles

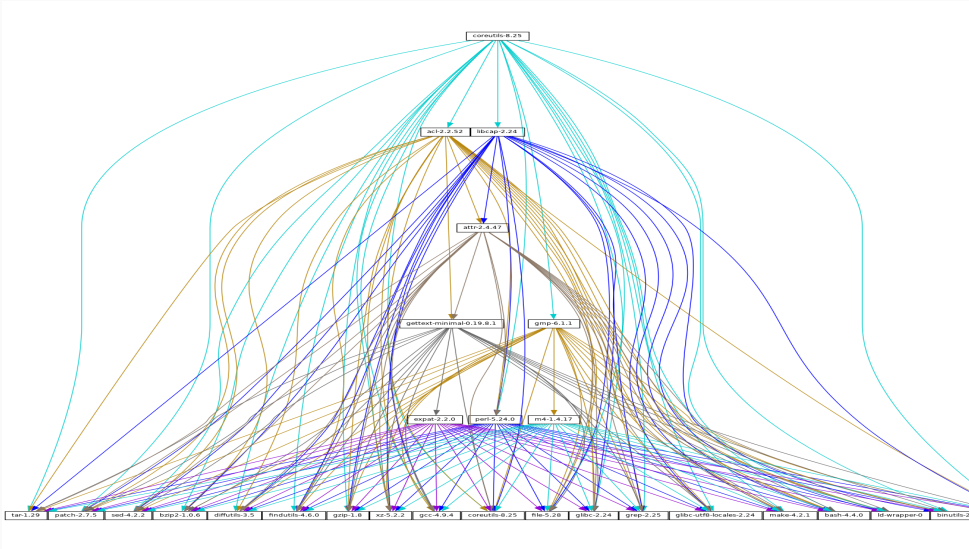
Développement de bibliothèques IP en langage VHDL / Verilog [7]

- Chaque (version) d'une IP dépend d'autres (versions) d'autres IPs
  - Chaque (version) d'une IP dépend d'autres (versions) d'autres IPs
    - Chaque (version) d'une IP dépend d'autres (versions) d'autres IPs
    - ...

Parallèle entre le logiciel ...



... et le hardware



7. hdl library: what is needed ?

- développement local
- simulation (nvc, ghdl, etc.)
- scripting (python, tcl, etc.)
- bibliothèques spécifiques (Python-Vunit, OSVVM, etc.)
- tests unitaires et intégration continue
- déploiement sur cluster
- partage de code
- retour en arrière en cas de bugs

On veut du déterminisme

```
git clone ...
make
# Done.
```

Toujours mêmes résultats par poste.

approche : custom channel

Repo git en ligne `$MYCHANNEL`

```
git clone $MYCHANNEL $GUIX_PACKAGE_PATH
git install -L $GUIX_PACKAGE_PATH ip-sdram-ctrl
```

```
(define-module (ip)
  #:use-module ((guix licenses) #:prefix
  ↪  license:)
  #:use-module (guix download)
  #:use-module (guix utils)
  #:use-module (gnu packages python)
  #:use-module (gnu packages cmake))

(define-public myip
  ...)
```

Usage d'un `build-system` générique customisable

```
(build-system copy-build-system)
(arguments
 (list
  #:install-plan ''(("." "share/osvvm/osvvm/" #:include ("vhd" "pro")))))
```

contrôleur SDRAM

```
(define-public ip-sdram-ctrl
  (let ((commit "b6a0387f...")
        (revision "1"))
    (package
      (name "ip-sdram-ctrl")
      (version
       (git-version "20200927" revision
       ↪  commit))
      ...
      (propagated-inputs
       (list ip-sdram-core)))))
```

dépendance IP

```
(define-public ip-sdram-core
  (package
    (name "ip-sdram-core")
    (version "1.0.1")
    ...
    (inputs (list nvc python ...))))
```

IP - contrôleur USB ft2232h

```
(define-public ip-ft2232h
  (package
    (name "ip-ft2232h")
    (version "0.1")
    (source
     (origin
       (method git-fetch)
       (uri ...)
       (file-name ...)
       (sha256 (base32 "1qm0....")))))
    (build-system copy-build-system)
    (arguments
     (list #:install-plan
      ''(("src" "share/ip/ft2232h/")
         ("sim"
         ↪  "share/ip/ft2232h/"))))
    (propagated-inputs
     (list ip-hpd
     ↪  ip-delay-substract))
    (home-page ...)
    (synopsis "Syn")
    (description "Desc.")
    (license license:gpl3+)))
```

Packaging d'IP [a]

- chaque ip vis sa vie dans son propre repo git
- les ip ne partagent pas de structure de dossiers
- forges indépendantes
- toute la complexité est reporté dans la bibliothèque de IPs
- ip liés par des rapports de dépendance
- tests d'ip
- langage d'extension pour plus de flexibilité

a. hdl library: proposal

en somme : reproductibilité matérielle

Git submodules ne suffisent pas

- architecture distribué
- modularité et flexibilité
- bibliothèques composable : plusieurs canaux
- gestion automatisée des dépendances
- notion de privacité d'IP

Guix

- outil activement développé
- déterminisme et reproductibilité
- gestion des dépendances
- profiles indépendant par projet
- création d'environnements / déploiement hpc / ci sur forge
- fall back vers SH et machine arrière
- per-profile roll-back vers des transactions plus anciennes

bibliography

M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, pp. 452–454, May 2016.

M. Kosinski, "Evaluating large language models in theory of mind tasks."

N. Shapira, M. Levy, S. H. Alavi, X. Zhou, Y. Choi, Y. Goldberg, M. Sap, and V. Shwartz, "Clever hans or neural theory of mind ? stress testing social reasoning in large language models," 2023.

M. Mitchell, "How do we know how smart ai systems are ?," *Science*, vol. 381, July 2023.

C. Reinhart and K. Rogoff, "Growth in a time of debt."

R. Ainsworth, "Reproducible and open science," 2022.

T. T. W. Community, "The turing way : A handbook for reproducible, ethical and collaborative research."

P. Dellaiera, "Reproducibility in software engineering," 2024.


R. D. Cosmo, "Archiving and referencing source code with software heritage," in *International Congress on Mathematical Software - ICMS 2020* (A. M. Bigatti, J. Carette, J. H. Davenport, M. Joswig, and p. u. family=Wolff, given=Timo, eds.), vol. 12097 of *Lecture Notes in Computer Science*, pp. 362–373, Springer, 2020.

Questions ?