

L'INTÉRÊT DE COUVERTURE DE TESTS AVANCÉE POUR BCI

Cas pratique BCIpy

ift 3913 angelo adragna 2024/2025

PROGRAMME

- Prérequis : couverture de tests avancée
- Concept de BCI
- Projet BCIPy de CAMBI
- Code, application pratique et généralisation
- Synthèse
- Devinette

PRÉREQUIS COUVERTURE DE TESTS AVANCÉS

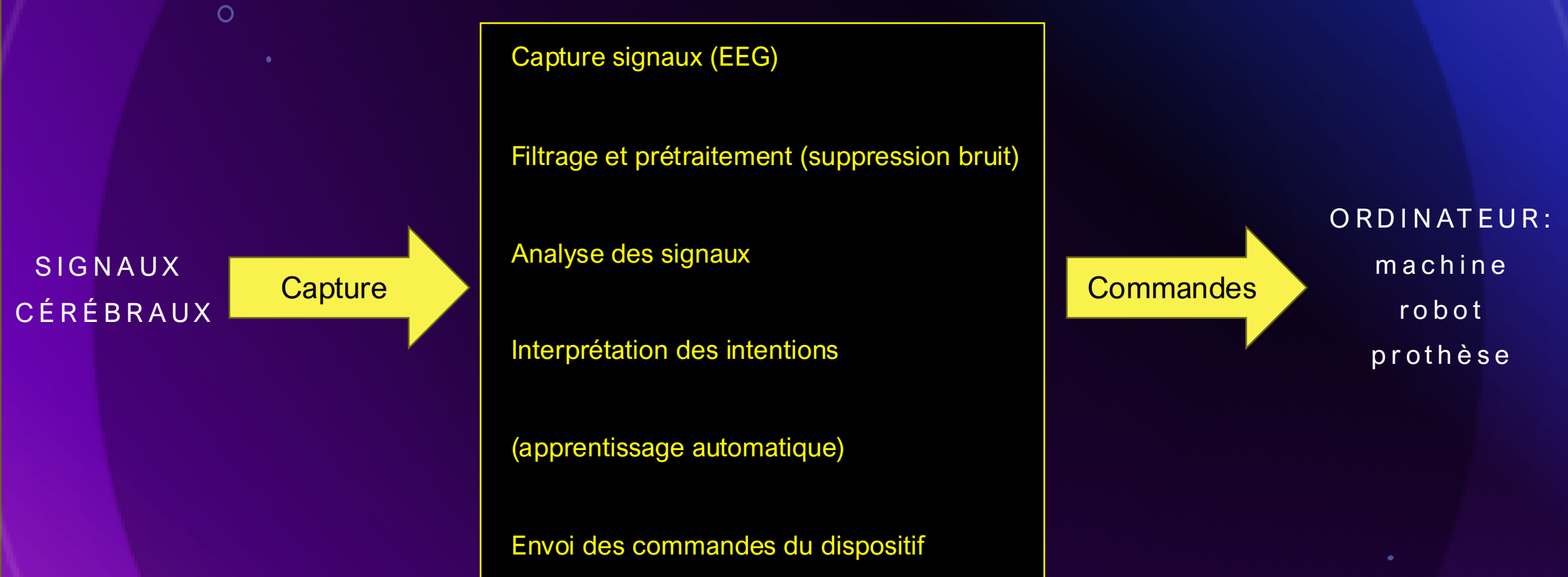
Couverture de
tests

&

Couverture de
tests avancés

- la couverture de code : le rapport entre nombre de lignes de codes exécutées par les tests et le nombre de lignes de codes total = Métrique et déterministe.
- L'analyse de mutation : Introduction variation du code censée provoquer une erreur pour montrer qu'un test est capable de détecter des erreurs = Mise en lumière de tests inefficaces.
- La couverture de scénarios : On vérifie que la plupart des scénarios sont testés. Scénario est un chemin fonctionnel ou cas d'utilisation.
- La couverture d'exigences : On vérifie que la plupart des exigences sont testées. Exigence est une description d'une fonctionnalité. Comme dans un cahier des charges.

VULGARISATION DU CONCEPT DE BCI (BRAIN COMPUTER INTERFACE)





Ref : <https://www.thedifferentia1dx.com>
C3PO gif

BCIpy par CAMBI

Groupe de chercheurs CAMBI.

Librairie fonctionnant comme application autonome et open source.

Réaliser des expériences d'interface cerveau-machine en python.

<https://github.com/CAMBI-tech/BciPy>

The screenshot shows the GitHub repository for BciPy, a Python Brain-Computer Interface Software. The repository is public and has 17 watchers, 33 forks, and 126 stars. It is maintained by the CAMBI-tech group. The repository includes a README, a license, and a code of conduct. The file list shows various files and folders, including .bciPy, .github, bciPy, scripts, .coveragerc, .gitignore, CHANGELOG.md, CODE_OF_CONDUCT.md, LICENSE.md, MANIFEST.in, Makefile, README.md, dev_requirements.txt, pytest.ini, requirements-winmac.txt, requirements.txt, setup.cfg, and setup.py. The repository is also linked to a website, bciPy.github.io, and has a list of releases, with the latest release being 2.0.1rc3 on May 2, 2023. The repository is also linked to a list of contributors, with 18 contributors listed.

BciPy Public

Watch 17 Fork 33 Star 126

main 62 Branches 15 Tags

Go to file Add file Code

About

Python Brain-Computer Interface Software

bciPy.github.io

python signal-processing data-acquisition python3 eeg language-model bci brain-computer-interface

Readme View license Code of conduct Activity Custom properties 126 stars 17 watching 33 forks Report repository

Releases 9

2.0.1rc3 Latest on May 2, 2023 + 8 releases

Packages

No packages published

Contributors 18

tab-cmd	Merge pull request #284 from CAMBI-tech/patch_rc3	02591eb · last year	2,084 Commits
.bciPy	Updates to make bciPy install standalone. Add missing inits	last year	
.github	Upgrade psychopy and add support for python3.9 (#272)	last year	
bciPy	Updates to make bciPy install standalone. Add missing inits	last year	
scripts	Home use testing: screen refresh alert and offline analysis ...	last year	
.coveragerc	more testing!	6 years ago	
.gitignore	Merge branch '2.0.0rc3' into backup	last year	
CHANGELOG.md	update changelog	last year	
CODE_OF_CONDUCT.md	Update license, code of conduct, move bci_main to make a ...	3 years ago	
LICENSE.md	Update license, code of conduct, move bci_main to make a ...	3 years ago	
MANIFEST.in	Updates to make bciPy install standalone. Add missing inits	last year	
Makefile	address PR comments and set python versions in setup.py	2 years ago	
README.md	Upgrade psychopy and add support for python3.9 (#272)	last year	
dev_requirements.txt	Add missing inits and lock pyglet / dev requirements (#258)	2 years ago	
pytest.ini	Prestimulus, Inquiry Based Training, Model Tuning (#208)	2 years ago	
requirements-winmac.txt	Upgrade psychopy and add support for python3.9 (#272)	last year	
requirements.txt	New default Wearable Sensing channels, upgrades to perm...	last year	
setup.cfg	lint	5 years ago	
setup.py	Updates to make bciPy install standalone. Add missing inits	last year	

Code, pratique et théorie : Couverture de scénarios

- Test "prédiction" de gpt2 sans contexte : liste vide en entrée, on récupère les probabilités puis on les analyse.

```
112 def test_gpt2_predict_start_of_word(self):
113     """Test the gpt2 predict method with no prior evidence."""
114     symbol_probs = self.gpt2_model.predict(evidence=[])
115     probs = [prob for sym, prob in symbol_probs]
116
117     self.assertTrue(
118         len(set(probs)) > 1,
119         "All values should not be the same probability")
120     # Consider whether the following assertion should be True
121     # backspace_prob = next(prob for sym, prob in probs if sym == BACKSPACE_CHAR)
122     # self.assertEqual(0, backspace_prob)
123     for prob in probs:
124         self.assertTrue(0 <= prob < 1)
125     self.assertAlmostEqual(sum(probs), 1, places=3)
```

- Pratique :

Ce code s'assure d'avoir les prédictions valides sans contexte au préalable. En pratique si les prédictions sans contexte étaient "parfaites" la main pourrait se resserrer par reflexe quand la personne sursaute.

- Théorie :

Sans couverture de scénarios, on entrave l'anticipation des comportements réels, tel que l'état mental et physiologique. Donc provoquer des mouvements incorrects ou non désirés.

Code, pratique et théorie : Couverture d'exigences

- Ce code test le constructeur CausalLanguageModel de gpt2 : pour s'assurer de lever une exception UnsupportedResponseType lorsque le type de réponse fourni n'est pas supporté par le modèle.

```
51 def test_unsupported_response_type(self):
52     """Unsupported responses should raise an exception"""
53     with self.assertRaises(UnsupportedResponseType):
54         CausalLanguageModel(response_type=ResponseType.WORD,
55                               symbol_set=alphabet(), lang_model_name="gpt2")
56
```

- Pratique :

Ce test contribue à la robustesse et à la fiabilité en garantissant que la classe réagit correctement aux entrées non valides et en prévenant les erreurs potentielles à l'exécution.

- Théorie :

Sans couverture d'exigences, on s'expose à l'imprécision, au manque de robustesse et aux fonctionnalités manquantes. Donc des mouvements manquants ou non pris en charge.

~~Code, pratique et~~ théorie : Analyse de mutation

- Théorie :

Sans analyse de mutation, on pourrait laisser passer des prédictions moins précises en grande partie à cause de tests inefficaces. Donc des mauvaises interprétations peuvent provoquer des mouvements saccadés (lag et latence).

SYNTHESE

- Exemples physiques et concrets : pour montrer intérêts de la couverture de code avancée.
- Plus qu'une métrique.
- Causer des réels problèmes, dans le BCI comme ailleurs.

Le BCI c'est quand même la classe !

DEVINETTE



<https://www.cambi.tech/about>

COMMENT RECONNAIT-ON DES SCIENTIFIQUES
SEULEMENT AVEC UNE PHOTO ?