Créer un chatbot pour réserver les vacances

FLY ME

Contexte

Fly Me est une agence qui propose des voyages clé en main pour les particuliers ou les professionnels.

Pour attirer de nouveaux types d'utilisateurs nous aimerions développer un chatbot capable d'aider nos clients à choisir une offre de voyage.

Nous avons donc préparé une première version que nous allons tester en interne en permettant à notre équipe d'y réserver des billets d'avions.

Objectif

Notre premier objectif était donc de réaliser un MVP permettant d'interagir via un outil de chat avec un agent conversationnel à intelligence artificielle développé sur la base d'un jeu de données représentatif.

Pour ce faire, nous avons suivi ces étapes :

- **EDA/Preprocess - analyse et préparation des données**
- **Modèle LUIS - préparation d'un modèle LUIS.ai**
- **Chatbot - création d'un chatbot interagissant avec LUIS**
- **Télémetrie - envoi d'informations sur Azure Insights**
- **CI/CD - déploiement automatique sur Azure Web-app**
- **Tests Unitaires - vérifications automatisées**

(pas forcement dans cet ordre...)

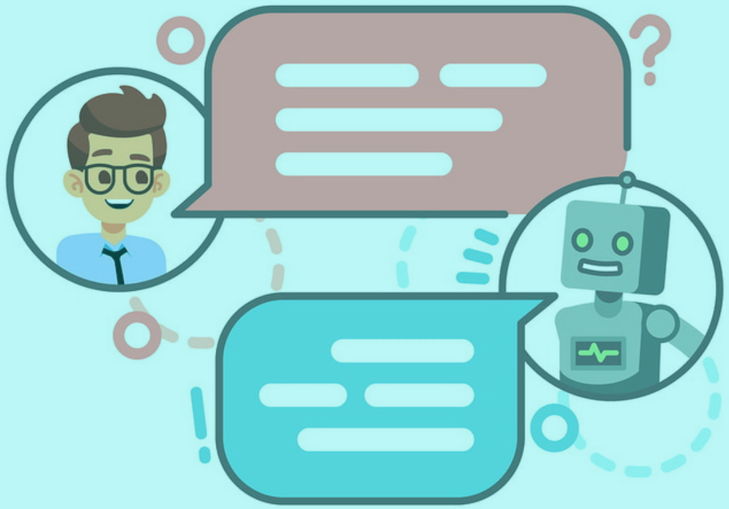
Le jeu de données

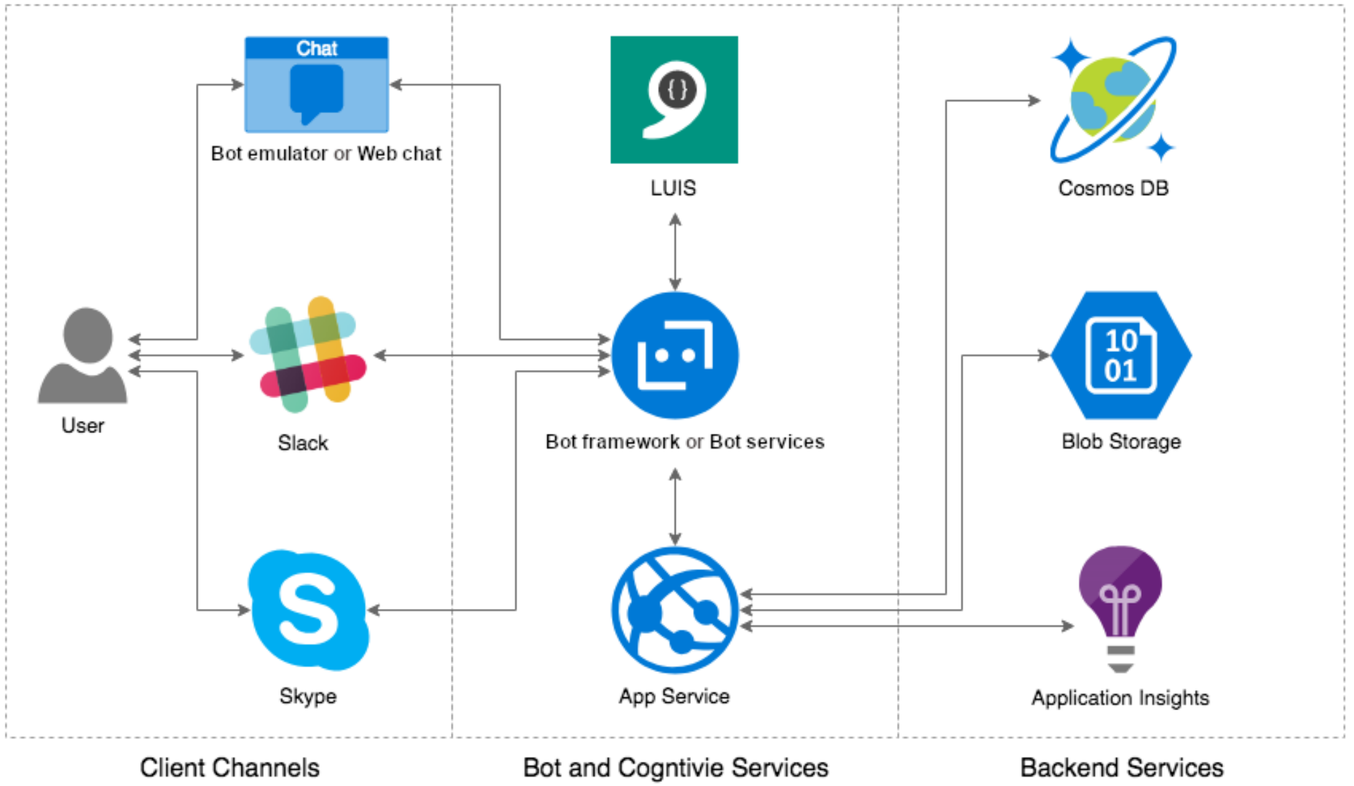
Nous avons travaillé avec un jeu de données venant de Microsoft et contenant des dialogues entre des clients voulant faire une réservation (vols, hôtels, restaurants...) et un service client.

Cet ensemble contient:

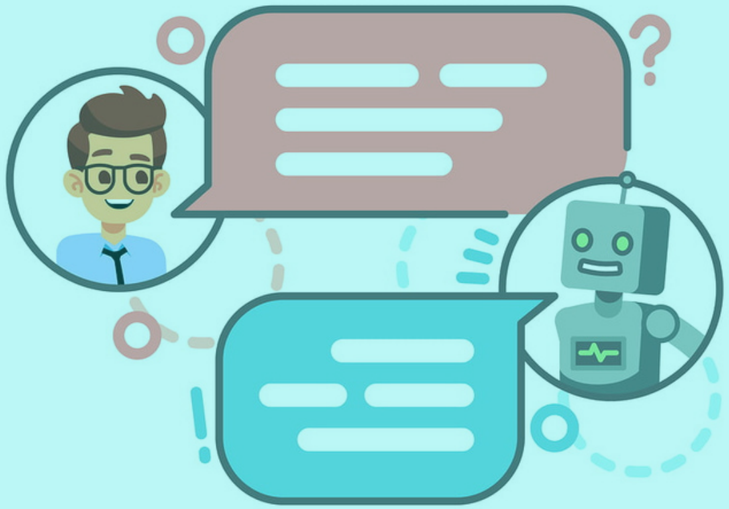
- 1369 dialogues complets entre 11 clients et 12 bots
- 19 986 échanges au total
- l'intention associée à chaque échange
- les entités identifiées avec leur position et valeur

Comment fonctionne un Chatbot ?



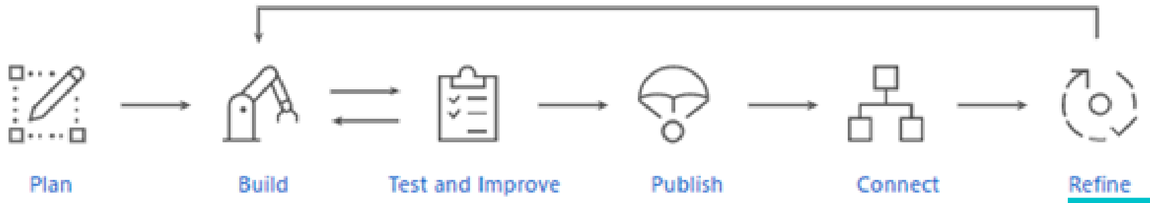


Modèle LUIS



Azure Language Understanding (LUIS) est un service cloud d'IA conversationnelle qui applique des algorithmes personnalisables aux textes qui lui sont envoyés pour prédire le sens général et extraire des informations pertinentes et détaillées.

Ce service est accessible via un portail ou une API qui permettent de préparer, entraîner, évaluer et publier facilement des modèles.



Modèle LUIS \ Évaluation à l'entraînement

LUIS étant in fine un système de classification, il existe de nombreuses métriques que nous pourrions utiliser sur un test-set, telles que l'Accuracy, la Precision, le Recall, la Specificity, le F-score, le ROC AUC ou le PR AUC...

Mais l'équipe de LUIS s'est concentré sur 3 d'entre elles

- **Precision : taux vrai-positifs parmi les positifs**
  (quelle part des intentions ou entités prédites sont les bonnes ?)
- **Recall : capacité à détecter les vrai-positifs**
  (quelle part des véritables intentions ou entités ont été effectivement prédites ?)
- **F1-Measure : moyenne harmonique entre Precision et Recall** -> 2*(P*R) / (P+R)

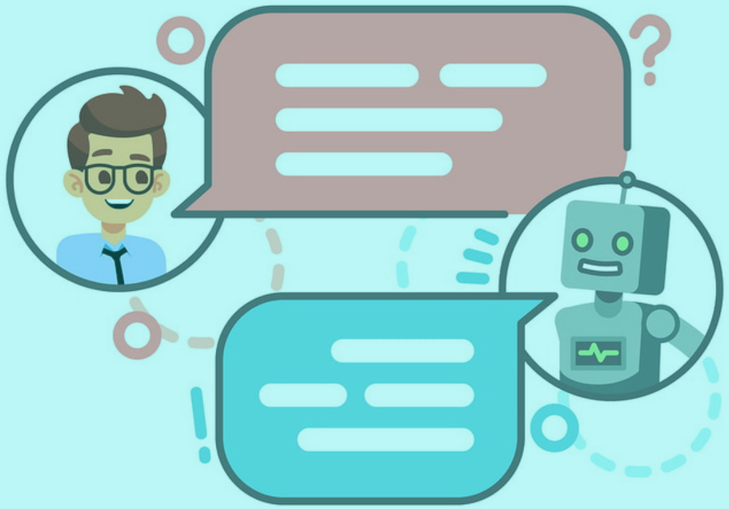
Modèle LUIS \ Démonstration

Scripts & Portail

python™

Azure LUIS API

Modèle LUIS \ Évaluation en production

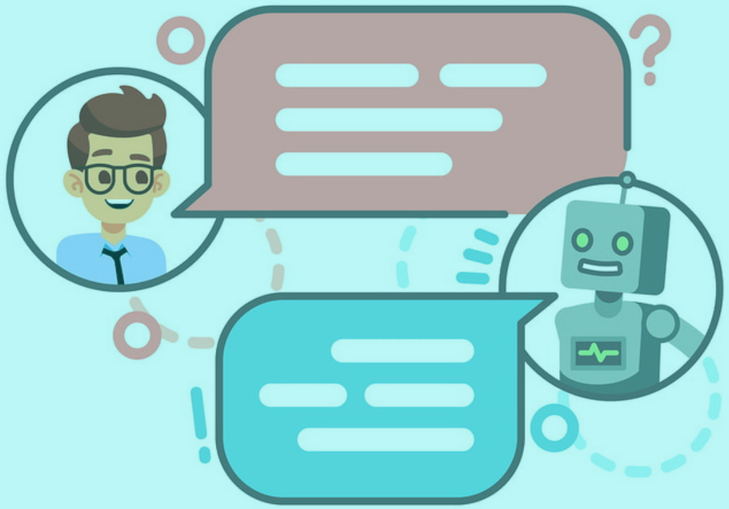


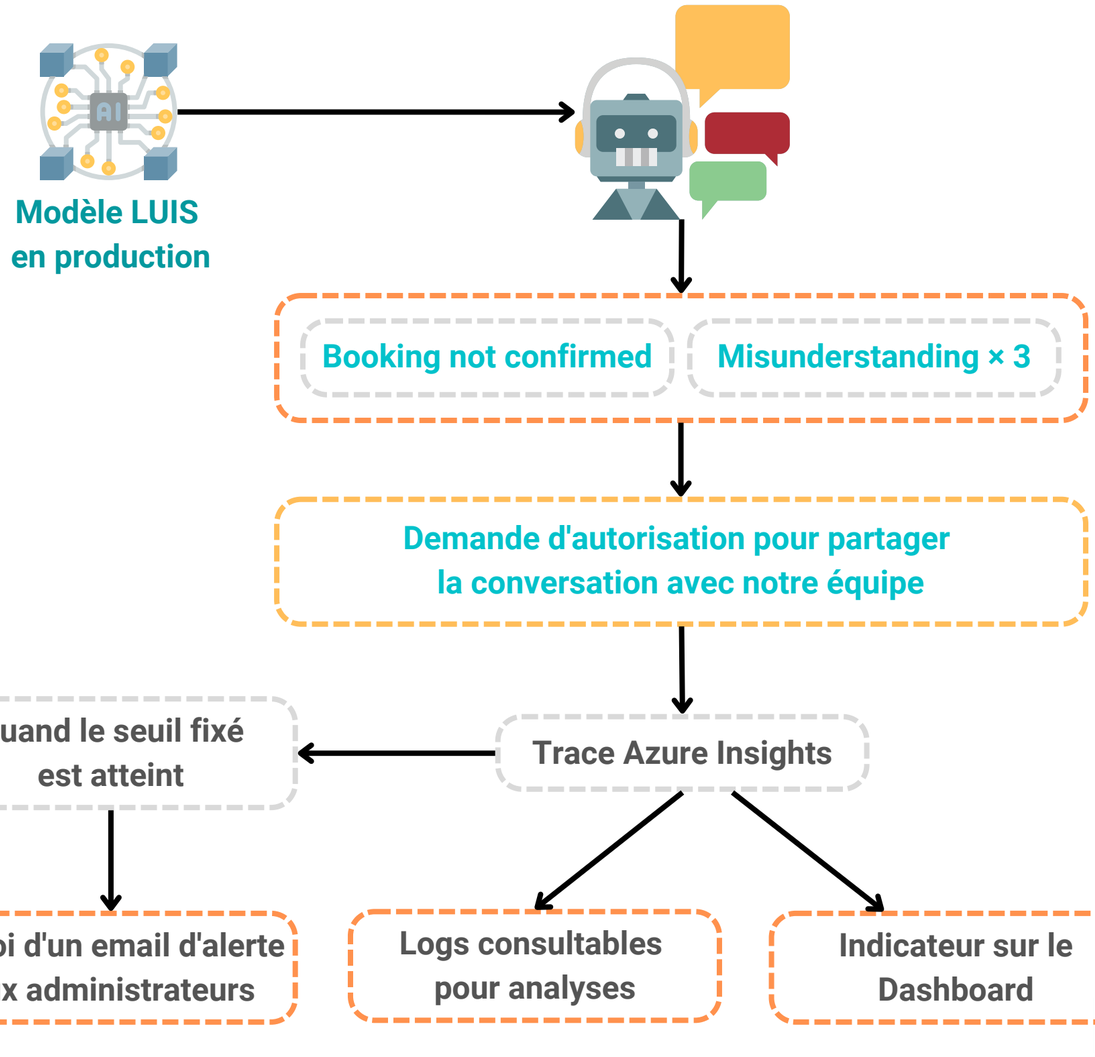
Il est difficile d'évaluer les classifications en temps réel.
Mais nous pouvons malgré tout essayer d'évaluer la qualité des dialogues sur la base de retours clients implicites ou explicites...

- demander l'avis des utilisateurs
- regarder le % de réutilisation du chatbot
- étudier le rapport entre nb de clients / nb d'achats
- lever des alertes dans certaines conditions

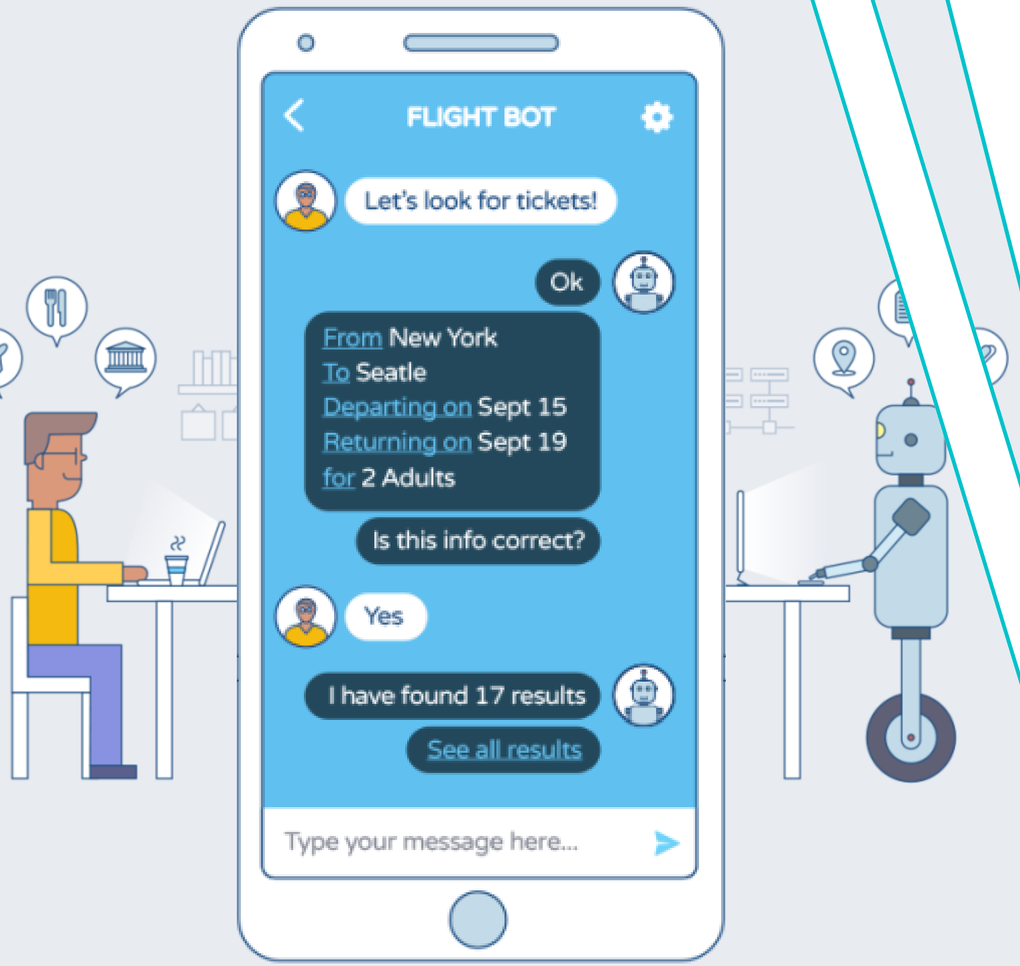
- Booking not confirmed
- Misunderstanding x 3

Modèle LUIS \ Évaluation en production





FlyMe Bot \ Déploiement



pytest

```
============================================ test session starts ============================================
platform linux -- Python 3.8.10, pytest-6.2.3, py-1.11.0, pluggy-0.13.1 -- /home/valkea/Dev/OpenClassrooms/Proje
ts_AI/P10/venvP10/bin/python
cachedir: .pytest_cache
rootdir: /home/valkea/Dev/OpenClassrooms/Projets_AI/P10
collected 10 items

unittests/bot_test.py::DialogTestClientTest::test_dialog_missing_budget PASSED                    [ 10%]
unittests/bot_test.py::DialogTestClientTest::test_dialog_missing_closeDate PASSED                 [ 20%]
unittests/bot_test.py::DialogTestClientTest::test_dialog_missing_destination PASSED               [ 30%]
unittests/bot_test.py::DialogTestClientTest::test_dialog_missing_openDate PASSED                  [ 40%]
unittests/bot_test.py::DialogTestClientTest::test_dialog_missing_origin PASSED                    [ 50%]
unittests/bot_test.py::DialogTestClientTest::test_dialog_step_by_step_full_NO PASSED              [ 60%]
unittests/bot_test.py::DialogTestClientTest::test_dialog_step_by_step_full_YES PASSED             [ 70%]
unittests/bot_test.py::DialogTestClientTest::test_dialog_step_by_step_full_YES_with_wrong_inputs PASSED [ 80%]
unittests/bot_test.py::DialogTestClientTest::test_dialog_step_by_step_straight_answers PASSED     [ 90%]
unittests/bot_test.py::DialogTestClientTest::test_init PASSED                                     [100%]

========================================== 10 passed, 1 warning in 12.79s ==========================================
```

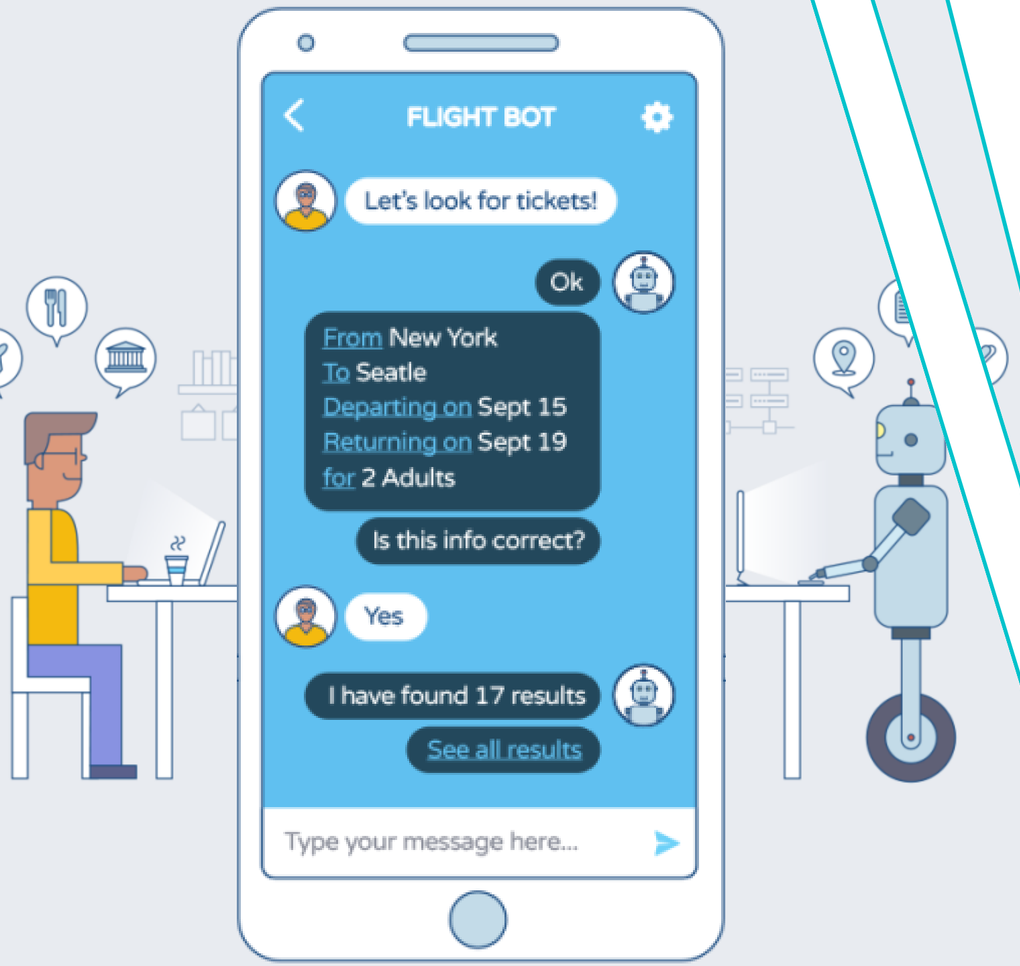
GitHub

GitHub Actions

build 2m 6s — deploy 4m 40s

https://flymyapp.azurewebsites.net

FlyMe Bot \ Démonstration



Démo
API / CLI

Bot framework

Azure LUIS

Azure Insights

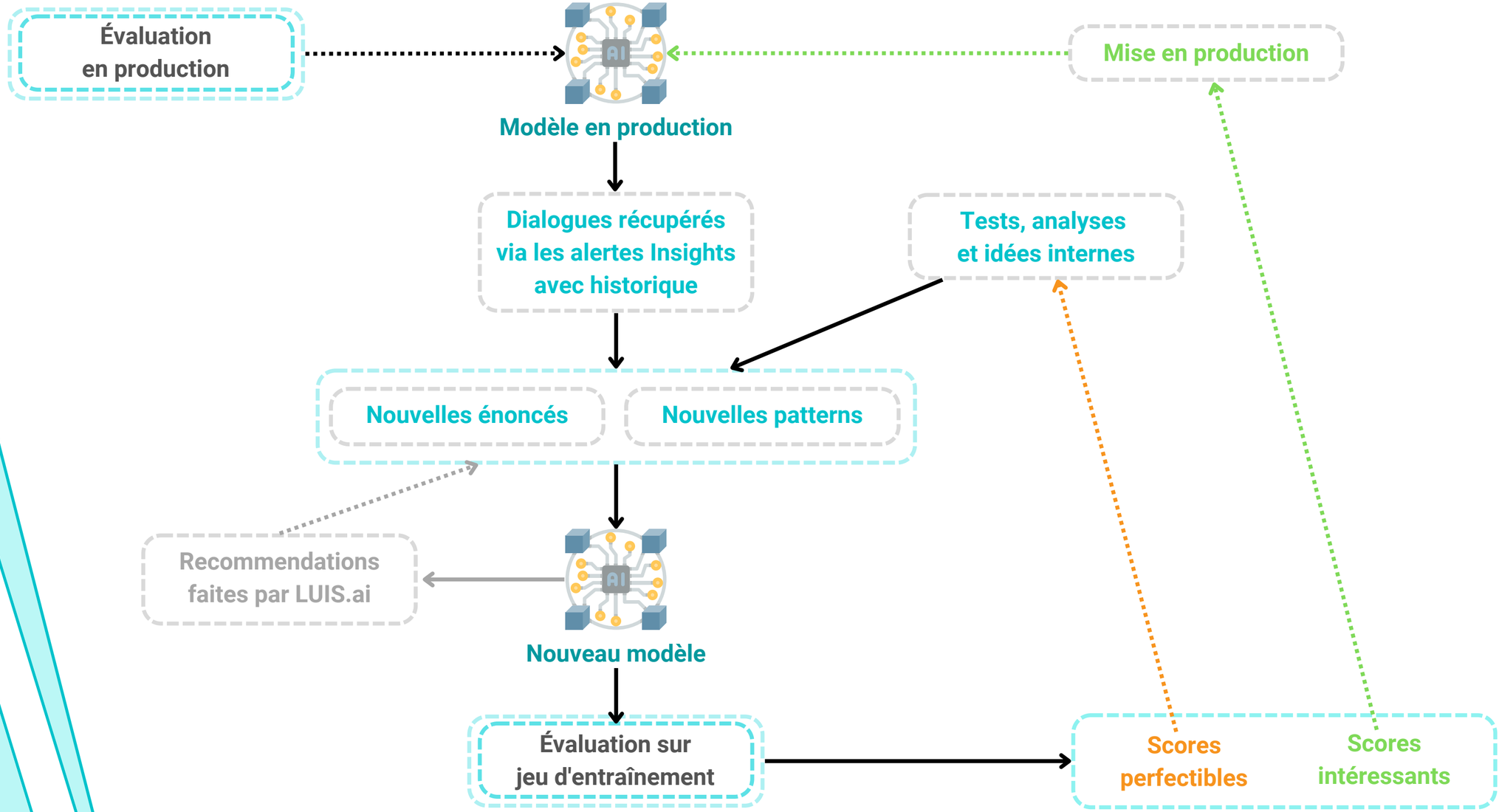
Azure Web-app

pytest

GitHub

GitHub Actions

Modèle LUIS \ Mises à jour du modèle



Axes d'amélioration

- **Utiliser toutes les énoncés du jeu d'entraînement**
  (le modèle actuel n'a été entraîné qu'avec les phrases d'accroches)
- **Utiliser les patterns sur LUIS**
  (les patterns aideraient le modèle pour la détection des entités)
- **Ajouter d'autres métriques en production**
  (% de dialogues allant au bout, % de retours, % de ventes)
- **Déterminer une fréquence de re-entraînement**
  (en fonction du nb. d'alertes & du nb. d'énoncés collectés)
- **Migrer LUIS sur CLU**
  (algorithmes SOTA, support multilingues intégré ...)

Merci de m'avoir écouté et évalué

FLY ME