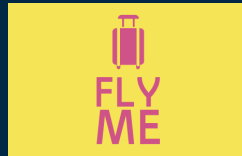


Parcours Ingénieur Intelligence Artificielle



- Projet 10 -
**Développez un chatbot pour
réserver des vacances**



Romain Le Goff
30/09/2022

Liens



GitHub : https://github.com/Deviluna29/oc_ingenieur-ia_P10

Chatbot :


url: <https://ocp10-flightbook.azurewebsites.net/api/messages>

App ID : 9cfd17e3-ffc4-45eb-b608-25e2ea31522a

App password : 37y8Q~U3wiGnKGwuwa4Lr1WG.8w4PjUgz1_6Gbko



Sommaire

- 
1. Contexte
 2. Présentation du jeu de données
 3. Service cognitif LUIS
 4. Chatbot
 5. Application Insights
 6. Architecture cible
 7. Conclusion



Contexte



- Fly Me est une agence qui propose des voyages clé en main pour les particuliers ou les professionnels. Fly Me a lancé un projet ambitieux de développement d'un chatbot pour aider les utilisateurs à choisir une offre de voyage.



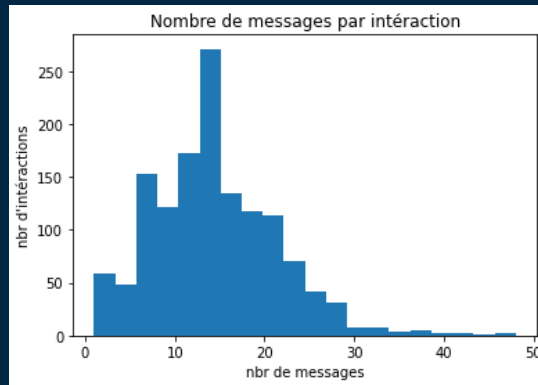
- Les objectifs :

- Utiliser un jeu de données disponible en ligne pour développer un MVP (V1 chatbot) qui aidera les utilisateurs à réserver un billet d'avion, et le déployer en production.
- Implémenter des tests unitaires pour s'assurer que le chatbot fonctionne correctement.
- Suivre et analyser l'activité du chatbot en production, pour détecter d'éventuelles mauvaises réponses.
- Mettre en place une méthodologie de mise à jour du modèle en production.

Présentation du jeu de données



- Le jeu de données contient des **historiques d'échange** entre un chatbot et un utilisateur :
 - **1369 échanges** composés d'en moyenne 15 messages
 - Informations pour chaque message : le texte, les labels associés à des éléments pertinents du texte, l'intention du message etc ..



Exemple de message d'un échange

```
{'text': "I'd like to book a trip to Atlantis from Caprica on Saturday, August 13, 2016 for 8 adults. I have a tight budget of 1700.",
 'labels': {'acts': [{'args': [{'val': 'book', 'key': 'intent'}],
   'name': 'inform'},
 {'args': [{'val': 'Atlantis', 'key': 'dst_city'},
 {'val': 'Caprica', 'key': 'or_city'},
 {'val': 'Saturday, August 13, 2016', 'key': 'str_date'},
 {'val': '8', 'key': 'n_adults'},
 {'val': '1700', 'key': 'budget'}],
   'name': 'inform'}]},
 'acts_without_refs': [{'args': [{'val': 'book', 'key': 'intent'}],
   'name': 'inform'},
 {'args': [{'val': 'Atlantis', 'key': 'dst_city'},
 {'val': 'Caprica', 'key': 'or_city'},
 {'val': 'Saturday, August 13, 2016', 'key': 'str_date'},
 {'val': '8', 'key': 'n_adults'},
 {'val': '1700', 'key': 'budget'}],
   'name': 'inform'}]},
 'active_frame': 1,
 'frames': [{'info': {'intent': [{'val': 'book', 'negated': False}],
   'budget': [{'val': '1700.0', 'negated': False}],
   'dst_city': [{'val': 'Atlantis', 'negated': False}],
   'or_city': [{'val': 'Caprica', 'negated': False}],
   'str_date': [{'val': 'august 13', 'negated': False}],
   'n_adults': [{'val': '8', 'negated': False}]}],
   'frame_id': 1,
   'requests': [],
   'frame_parent_id': None,
   'binary_questions': [],
   'compare_requests': []}],
 'author': 'user',
 'timestamp': 1471272019730.0}
```

Service cognitif LUIS



Avant de concevoir le chatbot, on va entraîner et déployer un **modèle d'analyse sémantique**, en utilisant un service d'Azure : le **service cognitif LUIS** (Language Understanding)

La V1 du chatbot doit identifier dans la demande de l'utilisateur 5 éléments :

- **Ville de départ, ville de destination, Date aller et date retour souhaitées des vols, budget**

Différentes étapes :

- Créer la ressource LUI (nom, description ..).
- Ajouter les **intents**, **entities** et **prebuilt entities** au modèle.
- **Formater le jeu de données**, pour le rendre compatible avec LUIS.
- Ajouter les exemples au modèle.
- **Entraîner** le modèle et le **déployer**.

- **or_city** (Ville de départ)
- **dst_city** (Ville de destination)
- **str_date** (Date aller souhaitée du vol)
- **end_date** (Date retour souhaitée du vol)
- **budget** (Budget maximum pour le prix total des billets)

```
{'text': "I'd like to book a trip to Atlantis from Caprica on Saturday, August 13, 2016 for 8 adults. I have a tight budget of 1700.",  
'intent_name': 'BookFlight',  
'entity_labels': [{'entity_name': 'dst_city',  
  'start_char_index': 27,  
  'end_char_index': 35},  
  {'entity_name': 'or_city', 'start_char_index': 41, 'end_char_index': 48},  
  {'entity_name': 'str_date', 'start_char_index': 52, 'end_char_index': 77},  
  {'entity_name': 'budget', 'start_char_index': 117, 'end_char_index': 121}]}
```

Test de l'appli LUIS

Exemple de requête :

I want to go to Paris with 250 dollars, leaving tomorrow

Réponse :

```
Top intent: BookFlight
Intents:
    "BookFlight"
Entities: {'dst_city': ['Paris'], 'geographyV2': [{'value': 'Paris', 'type': 'city'}], 'budget': ['250'], 'number': [250], 'str_date': ['tomorrow'], 'datetimeV2': [{'type': 'date', 'values': [{'timex': '2022-09-16', 'resolution': [{'value': '2022-09-16'}]}]}]}
```

Performance du modèle :

Jeu de test avec 100 échantillons -> precision de **71%**



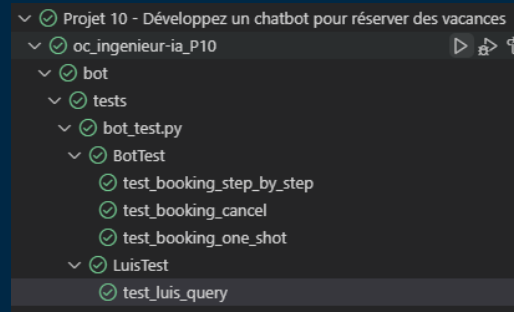
Chatbot



Pour construire le chatbot, on va utiliser le **Microsoft Bot Framework SDK**.

On va utiliser un exemple de base, que l'on va modifier / enrichir pour créer un bot correspondant à notre besoin :

- **Implémenter les dialogues** : message de bienvenue, demander des infos, confirmation, annulation ..
- **Connecter** le bot au **service LUIS**, pour sémantiser les messages de l'utilisateur.
- Le bot vérifie que les 5 éléments de la demande sont présents avant confirmation.
- Utiliser **Bot Framework Emulator** en local pour tester en communiquant avec le chatbot.
- Implémenter des **tests unitaire** (test de LUIS, test de dialogues)

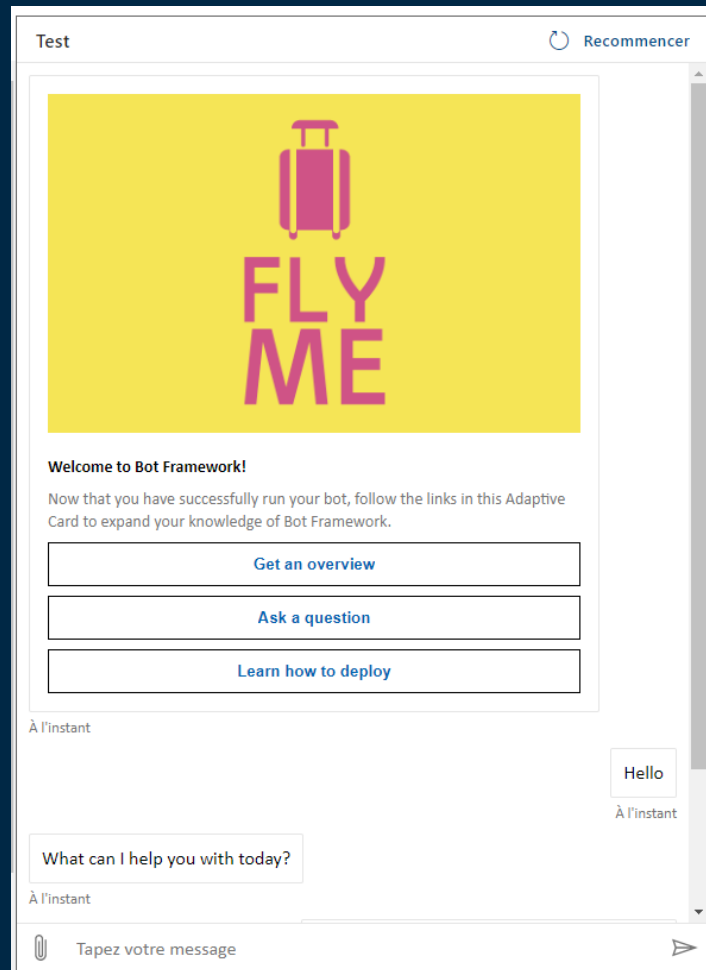


Déploiement du Chatbot

Déploiement sur Azure App



Utilisation d'Azure Bot



Application Insights



Afin de détecter d'éventuelles mauvaises réponses du chatbot, on va analyser son activité en production en utilisant un service Azure : **Application Insights**

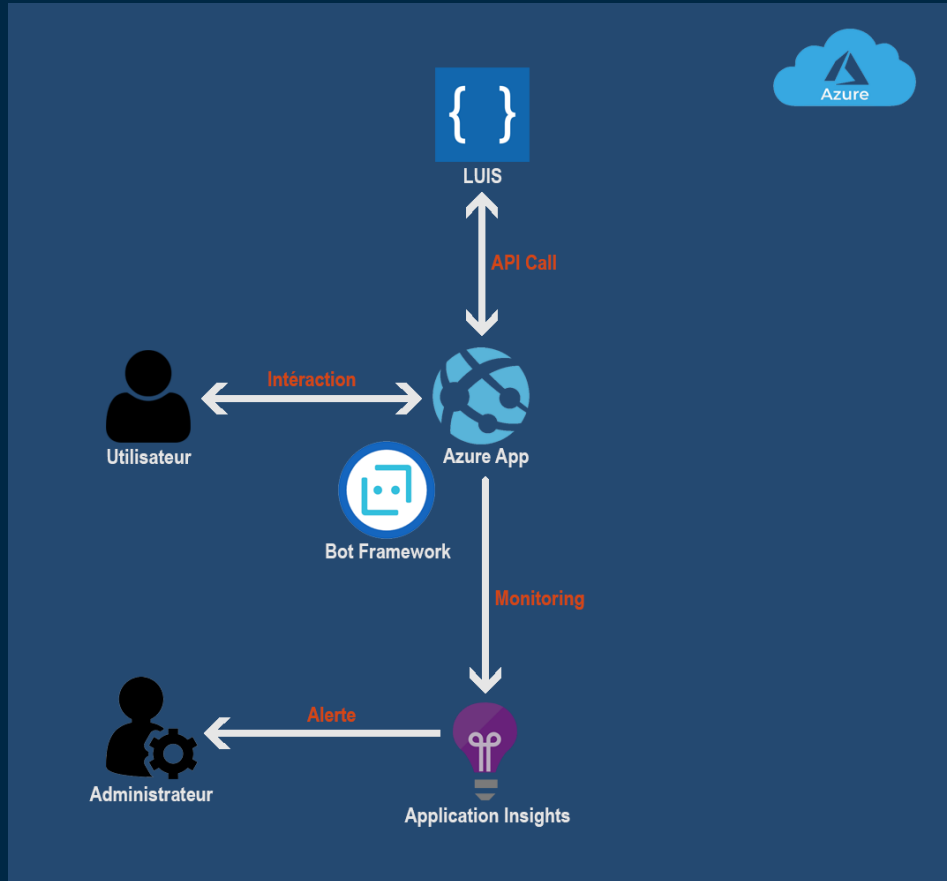
- On va surveiller 2 informations remontées par l'application (visible sur le dashboard) :

- **INFO** de niveau 1 (quand l'utilisateur **confirme** la proposition du bot)
- **ERROR** de niveau 3 (quand l'utilisateur **ne valide pas** la proposition)

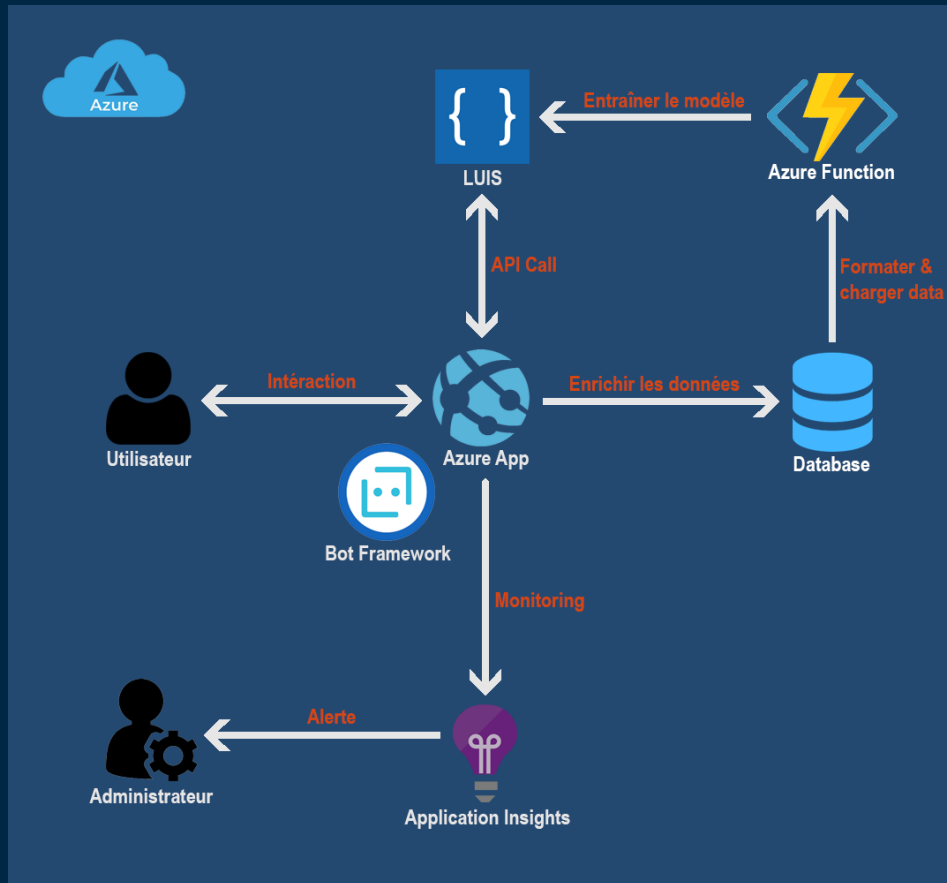
- Lorsqu'un utilisateur ne valide pas 3 fois une proposition dans un laps de temps de 5minutes, une **alerte** est remontée.



Architecture actuelle



Architecture cible



Cette architecture permet de :

- Récupérer les échanges entre utilisateurs et le bot, pour **enrichir le jeu de données** au fur et à mesure.
- Sur une **fréquence de 24h** par exemple, avec le service Azure Fonction, formater les données pour **ré-entraîner le modèle LUIS**.
- Evaluer la performance du nouveau modèle, si il est meilleur : le déployer.

Conclusion



Le chatbot a été conçu comme un **premier MVP**, il fonctionne bien et est une bonne base pour venir y greffer des améliorations :

- **Intégrer** le chatbot dans un channel comme Discord, Messenger ..
- Ajouter de nouvelles **entités** : nbr de passagers, nbr d'enfants, type de siège ..
- Augmenter la **précision du monitoring**, en remontant plus d'informations (erreurs, temps de réponses ..)
- Mettre en place une intégration continue (Architecture cible)

