

PROJET FLY ME

-

Méthodologie d'évaluation de la performance hors ligne, de pilotage en production et de mise à jour en production du modèle du chatbot



Introduction

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.

Pour cela, l'objectif du projet est de construire un MVP qui aidera les employés de Fly Me à réserver facilement un billet d'avion pour leurs vacances.

Ce document présente les éléments mis en place afin d'évaluer et suivre la performance du chatbot mis en place dans le cadre de ce projet.

Il est composé de 3 parties :

- 1 - Évaluation de la performance du modèle sous-jacent hors ligne
- 2 - Méthodologie de pilotage de la performance du modèle en production
- 3 - Méthodologie de mise à jour du modèle en production

1 - Évaluation de la performance du modèle sous-jacent hors ligne

Nous avons utilisé le service **Microsoft Azure LUIS** pour la modélisation. Le service LUIS permet d'une part de déterminer quelle est **l'intention de l'utilisateur** en fonction de sa demande et d'autre part d'identifier **les informations importantes** permettant de répondre à sa demande (*entities*).

Afin d'évaluer la performance du modèle, nous avons développé **des fonctions dédiées** dans le code Python permettant d'évaluer le modèle LUIS mis en place pour le chatbot.

Pour cela le jeu de données initial a été séparé en deux jeux de données :

- Un jeu de données pour **l'entraînement du modèle** (train)
- Un jeu de données pour **l'évaluation du modèle** (test)

Description des critères d'évaluation de la performance retenus :

Nous avons retenu les critères suivants pour l'évaluation de la performance du modèle :

- Pourcentage d'**intention correctement détectée** par le modèle pour l'intention **OrderTravel** (*réservation de vol*)
- Pourcentage d'**entités correctement détectées** pour l'intention **OrderTravel**
- Pourcentage d'**intention correctement détectée** pour l'intention **Greetings** (*salutations*)

Nous obtenons les résultats suivants :

- Pourcentage d'intention correctement détectée pour l'intention OrderTravel = **99%**
- Pourcentage d'entités correctement détectées pour l'intention OrderTravel = **84%**
- Pourcentage d'intention correctement détectée pour l'intention Greetings = **85%**

➔ **Ce sont de bons résultats qui montrent la qualité du modèle mis en place ainsi que sa bonne capacité de généralisation**

2 - Méthodologie de pilotage de la performance du modèle en production

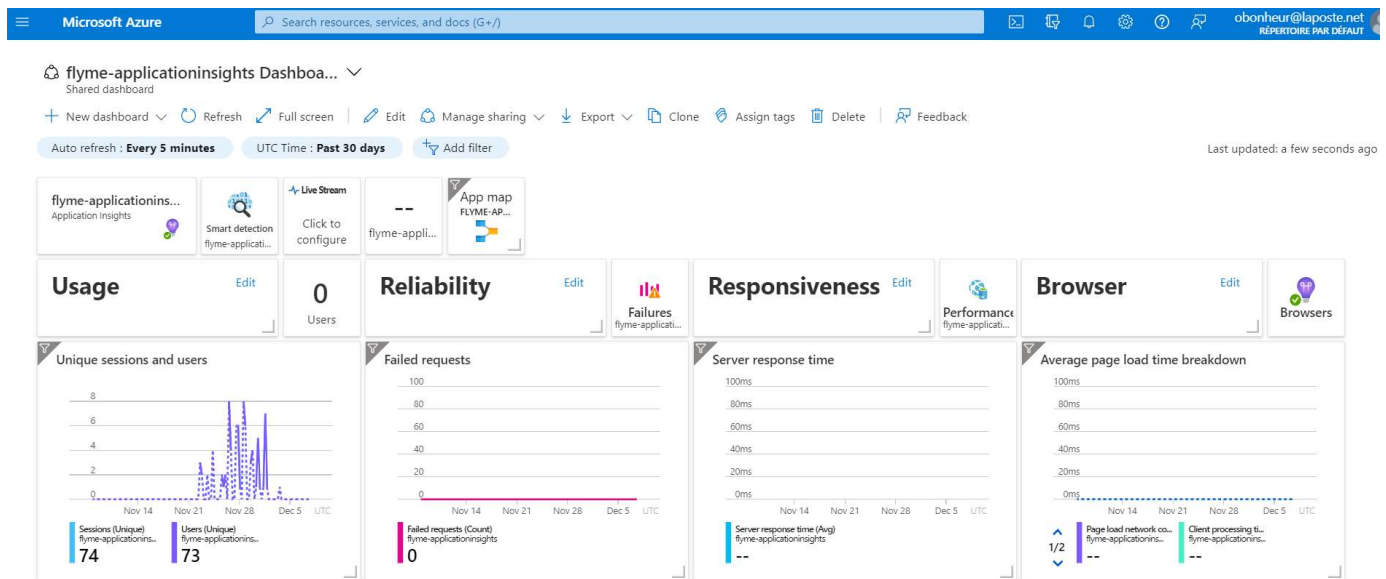
Pour le pilotage de la performance du modèle en production, nous avons utilisé le service dédié de Microsoft Azure : **Applications Insights**.

La mise en place se fait en 2 étapes :

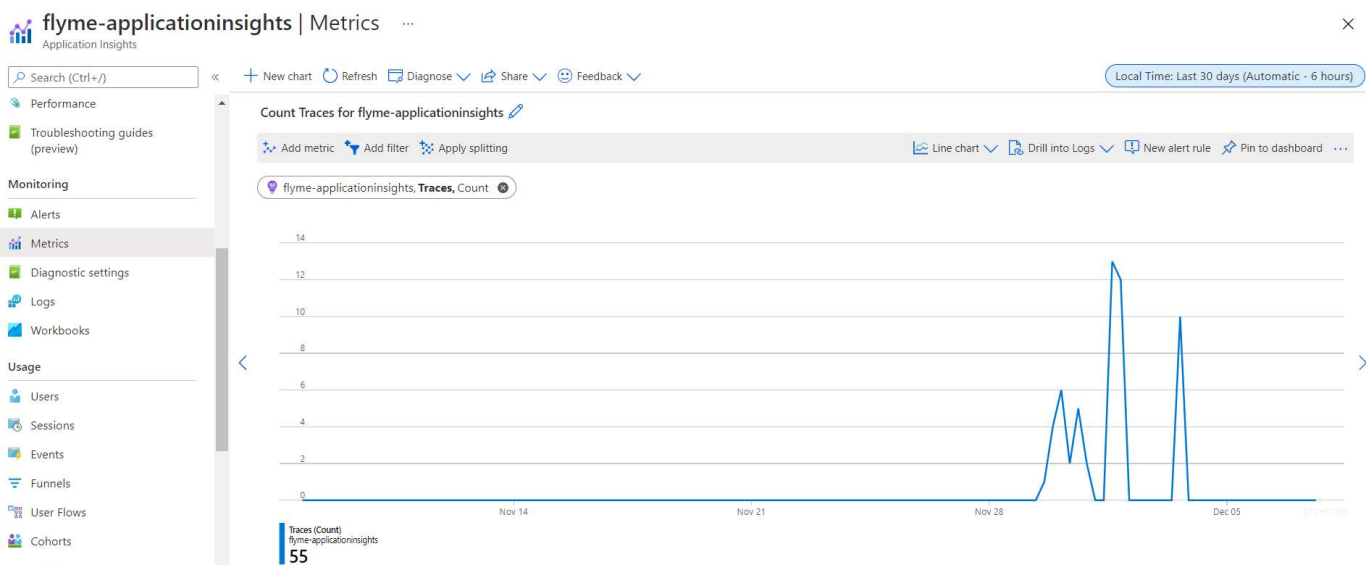
- On crée le service Applications Insights dans Azure
- On ajoute dans le code Python les éléments nécessaires permettant d'envoyer les informations au service Applications Insights

Applications Insights permet d'avoir accès aux services de suivi de la performance suivants :

- **Dashboard** : tableau de suivi récapitulatif personnalisable



- **Metrics** : métriques de suivi de la performance



- **Usage** : métriques de suivi de l'utilisation du chatbot
- **Transaction Search** : permet de suivre les événements majeurs qui se sont produits (Exceptions, Traces et étapes des échanges entre l'utilisateur et le chatbot notamment les labellisations effectuées par LUIS)
- **Availability** : permet de suivre la disponibilité de l'application
- **Alertes** : permet la mise en place d'alertes en cas de dégradation de la performance

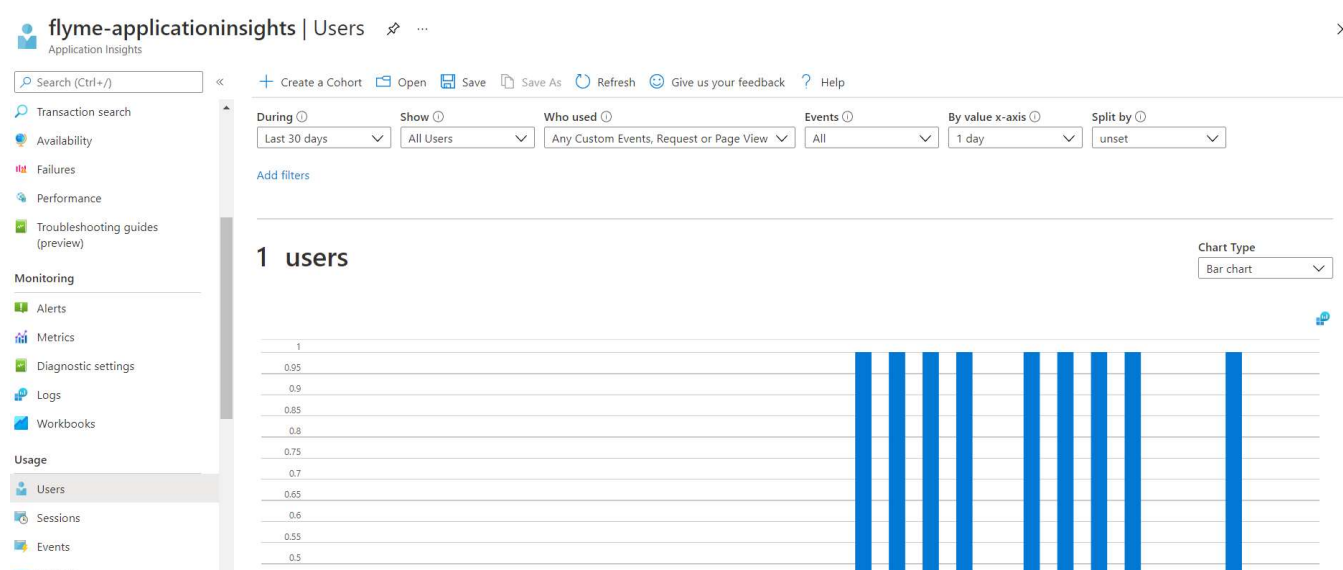
Description des critères de pilotage de la performance retenus :

Nous avons retenu les critères suivants pour le pilotage de la performance du modèle en production :

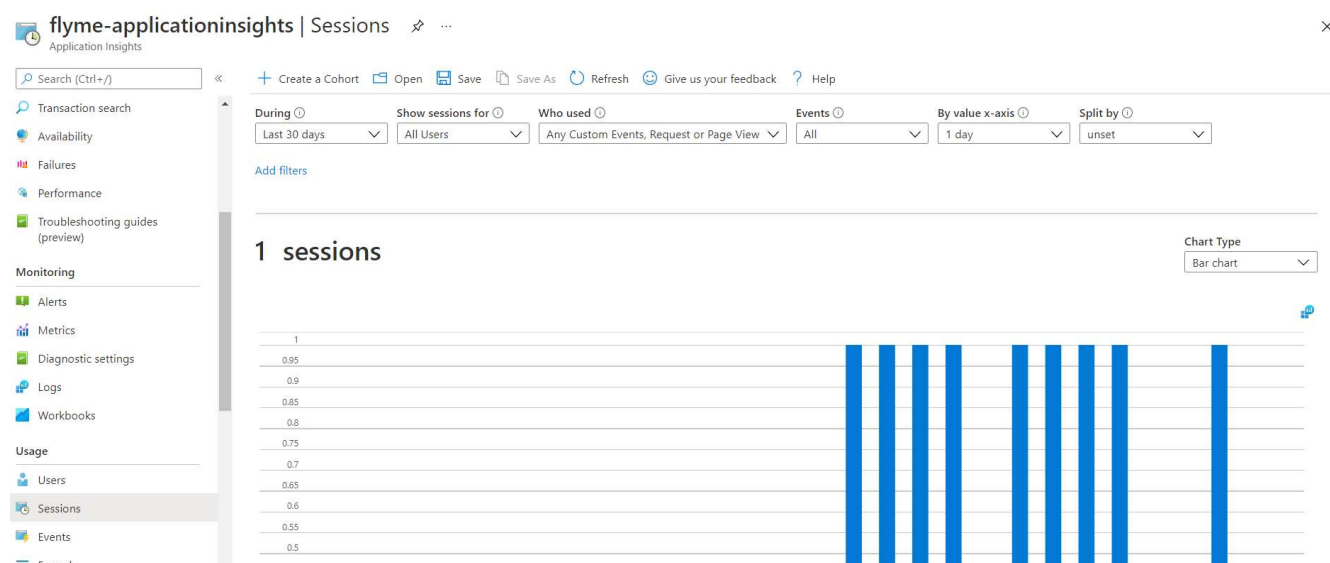
- Nombre d'utilisateurs sur l'application
- Nombre de sessions
- Disponibilité : mise en place d'un test de disponibilité
- Temps de réponse (**à voir une fois déploiement OK**)
- Dégradation de la performance : alerte si le chatbot fait 3 erreurs ou plus sur une période de 1 heure

Le nombre d'utilisateurs et le nombre de sessions sont calculés en continu par Azure et visibles dans l'onglet **Usage**.

Suivi du nombre d'utilisateurs :



Suivi du nombre de sessions :



Pour le suivi de la **disponibilité**, nous avons mis en place d'un test de disponibilité qui basé sur la réponse de l'application à un ping.

Mise en place du test de disponibilité :

The screenshot shows the 'Create test' dialog in the flyme-applicationinsights interface. The dialog is titled 'Create test' and has a close button (X) in the top right corner. It is divided into two main sections: 'Basic Information' and 'Test configuration'. The 'Basic Information' section includes fields for 'Test name' (with a placeholder 'Give your test a name'), 'SKU' (set to 'URL ping'), and 'URL' (with a placeholder and a help icon). Below these is a checkbox for 'Is your application private or behind a firewall?' with a 'Learn more' link. The 'Test configuration' section includes a checkbox for 'Parse dependent requests', a checkbox for 'Enable retries for availability test failures' (which is checked), and a 'Test frequency' dropdown set to '5 minutes'. At the bottom, there are three expandable sections: 'Test locations' (5 location(s) configured), 'Success criteria' (HTTP response: 200, Test Timeout: 120 seconds), and 'Alerts' (Enabled).

Résultat du test de disponibilité :

The screenshot shows the 'flyme-applicationinsights | Availability' page. The page has a sidebar on the left with navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Investigate (Application map, Smart detection, Live metrics, Transaction search, Availability), Failures, Performance, Troubleshooting guides (preview), and Monitoring (Alerts, Metrics). The main content area is titled 'Availability' and has a 'Local Time: Last 24 hours' filter. It displays a line chart showing availability percentage over time. Below the chart is a table titled 'Select availability test' with columns: AVAILABILITY TEST, 20 MIN, AVAILABILITY, and DURATION (AVG). The table shows one test: 'FlyMe-bot-availability' with an availability of 0.00% and a duration of 459 ms. To the right of the chart is a section titled 'Overall' with a bar chart showing 'Availability results' (Failed) with a count of 118. At the bottom right, there is a 'Drill into...' section with buttons for '0 Successful' and '118 Failed'.

Le **temps de réponse** est calculé par Azure et visible dans l'onglet **Metrics**.

Pour le critère de **dégradation de la performance** nous avons mis en place une alerte envoyant automatiquement un mail d'alerte si le chatbot fait 3 erreurs ou plus sur une période de 1 heure.

La mise en place de cette alerte a été réalisée en deux étapes:

- Dans le **code du chatbot** : utilisation de la méthode ***track_trace()*** permettant d'envoyer à Applications Insights le résultat des échanges avec l'utilisateur :
 - Si l'utilisateur a validé la bonne compréhension de sa demande par le chatbot : envoi d'une information de type **INFO** indiquant que **l'utilisateur a validé la transaction** (ainsi que la labellisation des messages par LUIS)
 - Si l'utilisateur n'a pas validé la bonne compréhension de sa demande par le chatbot : envoi d'une information de type **ERROR** indiquant que **l'utilisateur n'a pas validé la transaction** (ainsi que la labellisation des messages par LUIS)
- Dans **Applications Insights** : mise en place d'une règle avec les caractéristiques suivantes :
 - Calcul du nombre de fois où le chatbot a fait une erreur (c'est-à-dire que l'utilisateur n'a pas validé la transaction)
 - Si le nombre d'erreurs est supérieur ou égal à 3 sur une période de 1 heure, alors il y a un envoi automatique d'un message par email décrivant l'erreur

```
if step_context.result:
    self.telemetry_client.track_trace("Transaction confirmed by the user : YES", booking_info, "INFO")
    return await step_context.end_dialog(booking_details)

else:
    self.telemetry_client.track_trace("Transaction confirmed by the user : NO", booking_info, "ERROR")
    return await step_context.end_dialog()
```

Alerte mise en place dans Azure :

Dashboard > flyme-applicationinsights > Alert rules >

Alerte Erreurs Bot FlyMe

Edit alert rule

Save Discard Disable Delete Properties

Edit the details below to modify the alert rule.
When defining the alert rule, check that your inputs do not contain any sensitive content.

Scope

Select the target resource you wish to monitor.

Resource

flyme-applicationinsights

Edit resource

Condition

Configure when the alert rule should trigger by selecting a signal and defining its logic.

Condition name	Time
✓ Whenever the count of traces is greater than or equal to 3	1
Add condition	1

! You can add up to 5 conditions with a static threshold for a metric alert rule. All conditions must be met for an alert to be triggered.

Configure signal logic

Dimension name	Operator	Dimension values
Severity level	=	3
Select dimension	=	0 selected

! There are currently no values in the Severity level dimension. If values are added to this dimension in the future, the alert rule will monitor them.

Alert logic

Threshold

Static Dynamic

Operator: Greater than or equal to Aggregation type: Count Threshold value: 3 Unit: Count

Condition preview

Whenever the count of traces is greater than or equal to 3

Evaluated based on

Aggregation granularity (Period): 1 hour Frequency of evaluation: Every 5 Minutes

Mail reçu en cas d'alerte :

Azure: Activated Severity: 1 Alerte Erreurs Bot FlyMe

vendredi 3 Décembre, 16:22

De : Microsoft Azure

A : obonheur@laposte.net



⚠ Your Azure Monitor alert was triggered

Azure monitor alert rule Alerte Erreurs Bot FlyMe was triggered for flyme-applicationinsights at December 3, 2021 15:21 UTC.

Rule ID /subscriptions/dc0050bb-8e50-4b60-8aac-034371ba1a2a/resourceGroups/FLY-ME/providers/microsoft.insights/metricAlerts/Alerte Erreurs Bot FlyMe
[View Rule >](#)

Alerte visible dans Azure :

Alerte Erreurs Bot FlyMe

Alert details

Summary History Diagnostics

Severity
1 - Error

Fired time
12/3/2021, 4:21 PM

Affected resource
flyme-applicationinsights

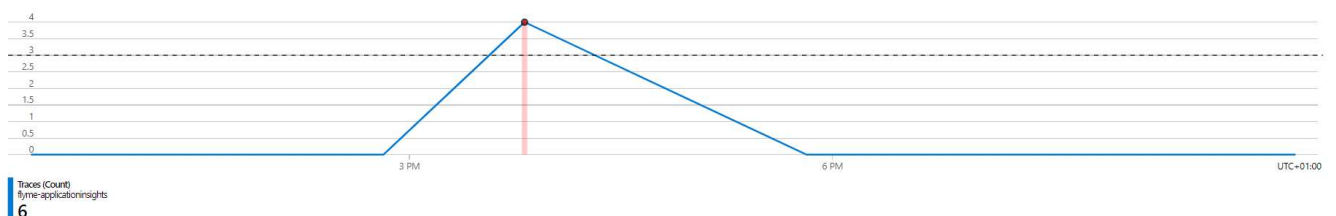
Hierarchy
Microsoft Azure Sponsorship 2 > fly-me

State
New

Monitor condition
Resolved

Change alert state

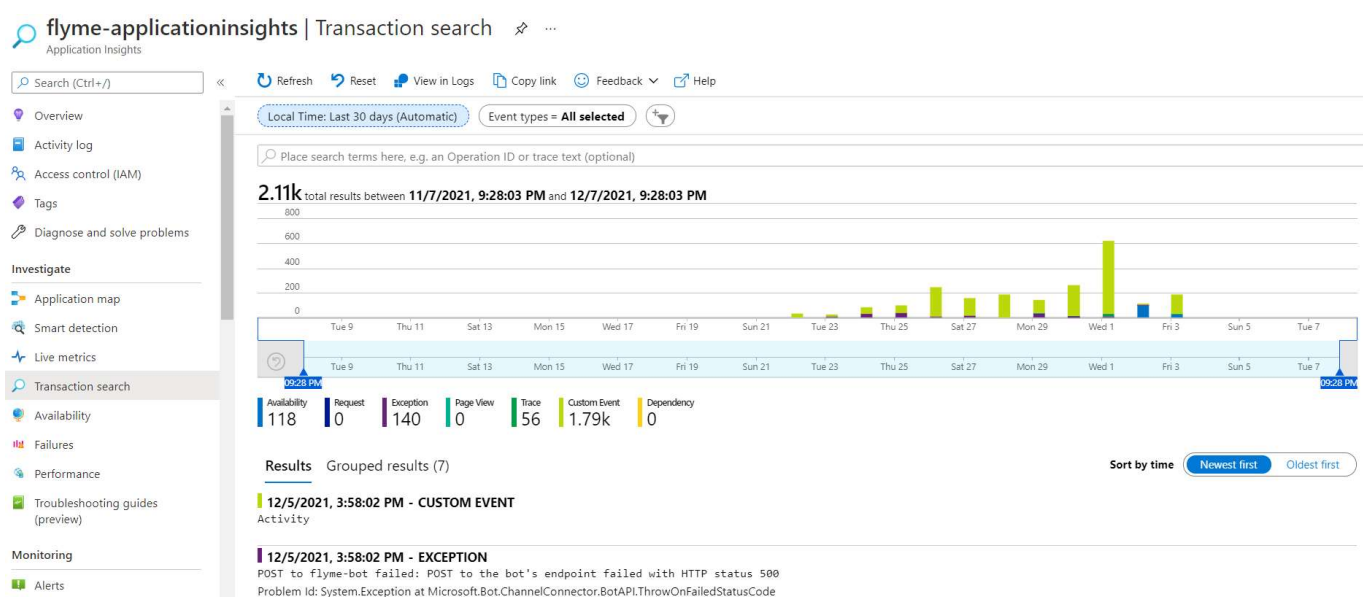
Why did this alert fire?



En plus de ces critères, le service **Transaction Search** permet de suivre les principaux événements liés à la vie du chatbot :

- **Exceptions :**
 - Erreurs au niveau du service (exemple : erreur HTTP 500)
- **Traces :**
 - Événements envoyés par le programme Python grâce à l'utilisation de la fonction ***track_trace()***
- **Custom Events :**
 - Événement envoyés automatiquement par le programme Python grâce à la classe ***BotTelemetryClient***, faisant partie du Framework SDK Bot de Microsoft
 - On retrouve notamment dans ces Custom Events les **échanges** et les **résultats labellisés** par LUIS

Tableau de suivi Transaction Search :



Exemple de suivi d'un échange en erreur (transaction non validée par l'utilisateur) :

End-to-end transaction details

flyme-applicationinsights

12/3, 4:19:34 PM - TRACE
Transaction confirmed by the user : NO
Severity level: Error

12/3, 4:19:34 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:19:29 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:19:25 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:19:25 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:19:25 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:19:22 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:19:17 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:19:17 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:19:17 PM - CUSTOM EVENT
LuisResult

12/3, 4:19:17 PM - CUSTOM EVENT
WaterfallStep

12/3, 4:02:15 PM - CUSTOM EVENT
WaterfallStep

Search results

Learn more

Copy link

Feedback

Leave preview

End-to-end transaction

1 All 1 Traces 0 Events

View timeline

Filter to a specific component and call

All [Component | Call]

ERROR 4:19:34.266 PM |

Transaction confirmed by the user : NO

Create work item

TRACE

Error

Trace Properties

Show all

Event time	12/3/2021, 4:19:34.266 PM (Local time)	...
Device type	Bot	...
Message	Transaction confirmed by the user : NO	...
Severity level	Error	...

Custom Properties

start_travel_date	2021-11-15T00:00:00.0000000Z	...
end_travel_date	2021-12-20T00:00:00.0000000Z	...
origin	par	...
destination	is	...
budget	3	...

Exemple de suivi des dialogues enregistrés et labellisés par LUIS :

End-to-end transaction details

flyme-applicationinsights

12/3, 3:58:40 PM - CUSTOM EVENT
LuisResult

12/3, 3:58:40 PM - CUSTOM EVENT
WaterfallStep

12/3, 3:58:40 PM - CUSTOM EVENT
WaterfallStep

12/3, 3:58:37 PM - CUSTOM EVENT
WaterfallStart

12/3, 3:58:37 PM - CUSTOM EVENT
WaterfallStep

12/3, 3:58:37 PM - CUSTOM EVENT
WaterfallStep

12/3, 3:58:37 PM - TRACE
Transaction confirmed by the user : NO
Severity level: Error

12/3, 3:58:37 PM - CUSTOM EVENT
WaterfallComplete

12/3, 3:58:37 PM - CUSTOM EVENT
WaterfallStep

12/3, 3:58:33 PM - CUSTOM EVENT
WaterfallStep

12/3, 3:58:33 PM - CUSTOM EVENT
WaterfallStep

12/3, 3:58:30 PM - CUSTOM EVENT
WaterfallStep

12/3, 3:58:30 PM - CUSTOM EVENT
WaterfallStep

Search results

Learn more

Copy link

Feedback

Leave preview

End-to-end transaction

1 All 0 Traces 1 Events

View timeline

Filter to a specific component and call

All [Component | Call]

CUSTOM EVENT 3:58:40.869 PM |

LuisResult

Create work item

CUSTOM EVENT

LuisResult

Custom Event Properties

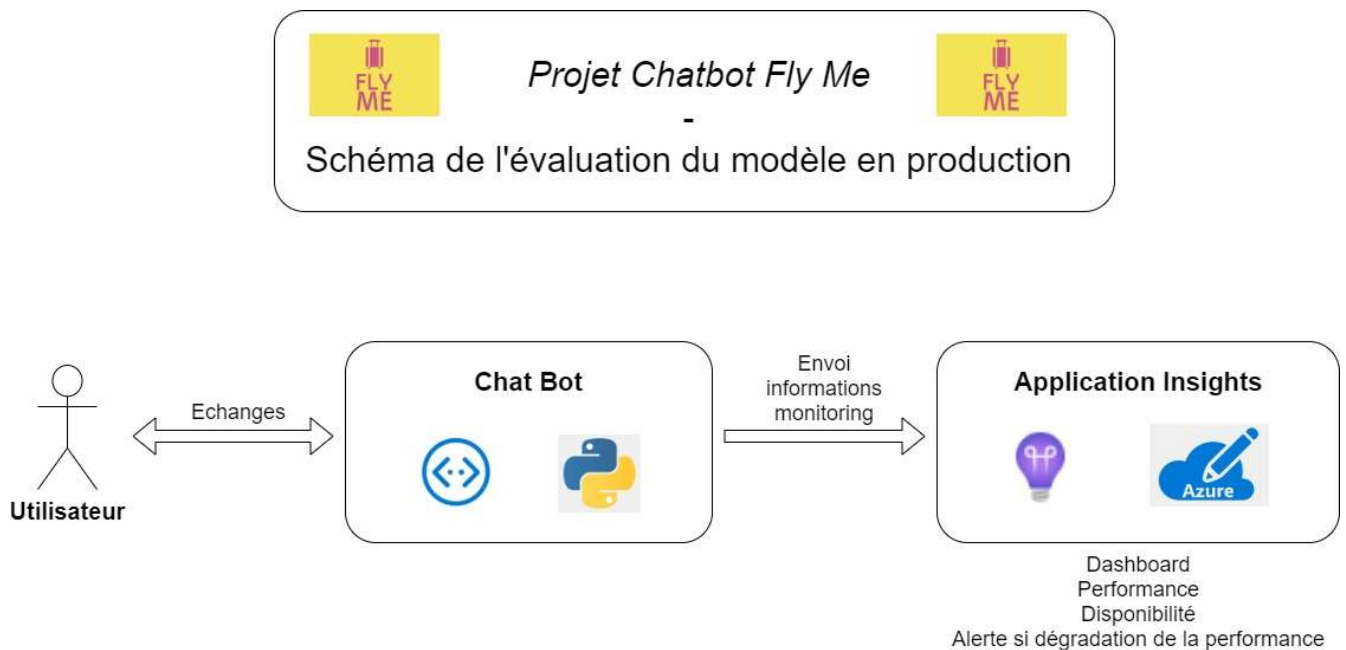
Show all

Event time	12/3/2021, 3:58:40.869 PM (Local time)	...
Event name	LuisResult	...

Custom Properties

applicationId	52c42e40-1395-47e1-9fcb-d8e59de2d71b	...
intentScore2	0.93	...
intentScore	0.93	...
entities	[{"instance": [{"ArrivalDate": [{"startIndex": 55, "endIndex": 71, "text": "20 december 2021", "type": "ArrivalDate", "score": 0.936669}], "DepartureDate": [{"startIndex": 35, "endIndex": 51, "text": "..."}]}]	...
intent2	OrderTravelIntent	...
fromId	4bbcc712-c65d-477e-bb17-eb49dc3372dd	...
intent	OrderTravelIntent	...

Schéma récapitulatif de l'évaluation du modèle en production :



3 - Méthodologie de mise à jour du modèle en production

La méthodologie proposée pour la mise à jour du modèle en production est la suivante :

1. Les données échangées entre l'utilisateur et le chatbot en production sont enregistrées
2. En cas d'un nombre d'erreurs trop importantes par le chatbot, le modèle est mis à jour :
 - a. Pour cela, on récupère tout d'abord les échanges préalablement sauvegardés
 - b. On procède à la labellisation de ces échanges (intention et entités)
 - c. Puis on ré-entraîne le modèle LUIS avec ces échanges en lançant le script dédié à cette tâche (`FLYME_SCRIPT-LUIS`)
3. On déploie le nouveau modèle
4. Et enfin, on observe le nouveau modèle en production :
 - a. S'il fait moins d'erreurs que le précédent on le garde
 - b. Sinon, s'il fait plus d'erreurs que le précédent, on revient au précédent modèle

➔ Cela permet d'essayer d'améliorer le modèle avec de nouveaux échanges, c'est-à-dire de nouvelles données, et de changer le modèle uniquement si le nouveau est meilleur que le précédent.