h e g

Haute école de gestion
Genève

L'architecture d'un chatbot intelligent au travers des solutions cognitives de Microsoft

EVOLUSYS
FROM INFORMATION TO RESULTS

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Nelson Farinha

Directeur de mémoire :

Philippe Dugerdil, Professeur HES

Genève, le 08 Juin 2018

Haute École de Gestion de Genève (HEG-GE)

Filière informatique de gestion

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre « Bachelor of Science en informatique de Gestion ».

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : http://www.urkund.fr/student_gorsahar.asp.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 08 Juin 2018

Nelson Farinha

Remerciements

Je souhaite remercier mon école la HEG (Haute Ecole de Gestion) ainsi que les professeurs m'ayant soutenu durant c'est 3 années de formation.

Je remercie aussi toute l'équipe du pôle Advanced Cloud Solutions d'Evolusys S.A qui m'a beaucoup apporté professionnellement et beaucoup aidé concernant les pistes de réflexion à suivre. Je souhaite remercier particulièrement Monsieur Etienne Marie qui m'a engagé chez Evolusys, me permettant de réaliser un travail de mémoire très passionnant.

J'aimerais remercier mon directeur de mémoire, Monsieur Philippe Dugerdil qui m'a apporté ses conseils, son engagement, son enthousiasme, son expérience et ses connaissances tout au long de ce mémoire.

Je remercie profondément ma mère et mes amis pour leurs incroyable soutien.

Résumé

L'idée de ce Travail de Bachelor en collaboration avec l'entreprise Evolusys est de trouver un moyen de présenter l'entreprise Evolusys à ses clients. Pour cela nous avons réfléchi avec Monsieur Etienne Marie, manager du pôle Advanced Cloud Solutions chez Evolusys, à une solution innovante sur le marché et qui parle à nos clients. La solution en question devra dans un futur proche, être implémentée chez nos clients. Nous avons donc décidé de réaliser un chatbot intelligent qui pourra répondre aux clients sur des questions en lien avec l'entreprise.

A travers cette recherche, nous allons apprendre à utiliser les différents services cognitifs pour réaliser un chatbot intelligent dans l'environnement de Microsoft cloud.

Nous présenterons par la suite notice de construction d'une architecture basée sur la technologie Microsoft permettant la construction d'un chatbot avec services cognitifs.

Pour finir, nous qualifierons la qualité des réponses aux questions générées par le chatbot sur l'entreprise.

Table des matières

1. Introduction 8
2. LUIS 9
2.1 Les différents concepts de LUIS 9
2.1.1 Une intention : 9
2.1.2 Un énoncé : 10
2.1.3 Une entité : 10
2.2 Accès à LUIS 10
2.3 Créez votre modèle LUIS 10
2.3.1 Identifier les entités 11
2.3.1.1 Entités prédéfinies : 11
2.3.1.2 Entités personnalisées : 11
2.4 Comment améliorer les performances de LUIS 11
2.4.1 Apprentissage actif : 12
2.4.2 Listes de phrases : 12
2.4.3 Patterns : 12
2.5 Configuration 12
2.5.1 Introduction à l'interface de LUIS 13
2.6 Capacités de LUIS au travers d'un jeux de tests 17
2.6.1 Construction des modèles : 18
2.6.2 Tests 20
2.6.2.1 Conclusion test : 21
2.7 Résumer 22
3. QnA Maker 23
3.1 Fonctionnement de QnA Maker 23
3.2 Configuration de QnA Maker 23
3.3 Résumer 25
4. Fiche technique Ms Translator 26
4.1 Fonctionnalités de Translator 26
4.2 Résumer 26
5. Bot Framework 27
5.1 Installation de l'environnement 28
5.2 Résumer 32
6. Architecture 33
6.1 Explication de l'architecture 34
6.2 Résumer 35
7. Analyse du chatbot 36

7.1 Jeu de tests 36
7.1.1 Explication des tests 37
8. Conclusion 38
Sitographie 39
Annexe 1 : Analyse des risques 40

Liste des tableaux

Tableau 1 : Tableau de test pour LUIS 20
Tableau 2 : Jeu de test final 36

Liste des figures

Figure 1 : Barre de navigation de LUIS 11
Figure 2 : Résultat de test de LUIS pour la phrase « I would like make an appointment with nelson farinha » 13
Figure 3 : Résultat de test de LUIS pour la phrase « take me an appointment with jean » 13
Figure 4 : Page d'accueil pour la création d'une nouvelle application LUIS 14
Figure 5 : Page de création d'une nouvelle intention 14
Figure 6 : Zone de nommage d'une nouvelle intention 14
Figure 7 : Page d'ajout des phrases clé pour une intention 15
Figure 8 : Page de création d'une nouvelle entité 15
Figure 9 : Zone de création d'une entité 16
Figure 10 : Modèle de LUIS avec ajout d'une entité 16
Figure 11 : 16
Figure 12 : Panneau de test de LUIS 17
Figure 13 : Intention de LUIS 18
Figure 14 : Model Appointment.Add 18
Figure 15 : Model Appointment.Delete 18
Figure 16 : Model Appointment.Find 19
Figure 17 : Exemple d'entrainement de LUIS 21
Figure 18 : Ajout d'une entité dans une phrase d'un modèle 21
Figure 19 : Panneau de test LUIS 22
Figure 20 : Page d'accueil de création d'un nouveau service de QnA Maker 24
Figure 21 : Page de nos question réponses 24
Figure 22 : Page de test de QnA Maker 25
Figure 23 : Page de publication de QnA Maker 25
Figure 24 : Canaux compatible avec Bot Framework 27
Figure 25 : Chat avec cartes adaptives personnalisé 27
Figure 26 : Microsoft Azure création d'un projet chatbot 28
Figure 27 : Elément d'un projet chatbot sur Azure 28
Figure 28 : Sélection du type de projet Bot Application 29
Figure 29 : Contenu du projet chatbot après sa création 29
Figure 30 : Code de la classe MessagesController.cs 30
Figure 31 : Procédure MessageReceivedAsync de la classe RootDialog.cs 30
Figure 32 : classe Web.config avec les identifiant masqué 31
Figure 33 : Page d'ajout de canaux de communication pour notre chatbot dans azure 32
Figure 34 : Architecture du chatbot 34
Figure 35 : Logo de Donna 38

1. Introduction

Les chatbots ne sont pas un phénomène récent et existent au travers de divers déclinaisons depuis de nombreuses années. Depuis peu, l'émergence de l'intelligence artificielle ouvre des perspectives encore inexploitées. Cette technologie qui évoque à la fois de l'espoir et des craintes. Les développeurs ont été capable de rendre les composants logiciels « plus intelligents ».

Grâce aux chatbot intelligents, le travail des collaborateurs dans leur entreprise va être facilité. Par exemple un chatbot pourra s'occuper des démarches internes d'une entreprise. Un autre s'occupera du service de support. Ils peuvent répondre à des milliers de cas différents. Dans ce travail, le chatbot concerné en est un qui pourra répondre à des utilisateurs qui poseront des questions en lien avec l'entreprise Evolusys.

Il existe mille et une technologies sur le marché pour réaliser des chatbot intelligents comme notamment Watson Assistant, la solution proposée par IBM, ou encore Lex qui est la solution proposée par Amazon. Pour ce travail de recherche, nous nous concentrerons sur la création d'un chatbot basé sur la technologie Microsoft. Le choix de la technologie s'est opéré au travers de deux leviers. Le premier est que cette technologie offre un large spectre de capacités pour l'utilisateur final mais également via l'expertise de l'entreprise Evolusys au travers de ses partenariat gold.

Dans ce travail, nous allons apprendre à utiliser les différents composants essentiels à la réalisation du chatbot intelligent dans l'environnement de Microsoft. Nous traiterons ces composants séparément pour ensuite les lier ensemble.

Pour finir, un jeu de test sera effectué pour estimer si le chatbot correspond à nos attentes. Les tests seront des questions sur l'entreprise Evolusys. Par la suite, nous analyserons la pertinence des réponses et les comparerons avec le résultat attendu.

2. LUIS

LUIS [1]est un service de compréhension de la langue proposé par Microsoft. Il permet aux applications de comprendre ce qu'une personne veut exprimer dans ses propres mots (en langage naturel). LUIS utilise l'apprentissage automatique pour permettre aux développeurs de créer des applications qui reçoivent en entrée des phrases en langage naturel puis en extraient le sens.

Une application LUIS est basée sur un modèle de langage spécifique à un domaine conçu par un développeur, un modèle de langage prédéfini par LUIS, ou un mélanger des deux.

Un modèle LUIS commence par une liste d'intentions générales de l'utilisateur, telles que *« Rendez-vous »* ou *« Contacts ».* On peut créer autant d'intentions que l'on a besoin. Une fois les intentions identifiées, il faut fournir des exemples de phrases appelées *« énoncés »* pour les intentions (il est recommandé de saisir au minimum 5 phrases). Ensuite il faut étiqueter les énoncés avec des détails spécifiques pour que LUIS les retire de l'énoncé qui sont appelés des Entités.

Les modèles de domaine prédéfinis incluent déjà toutes ces briques et constituent un excellent moyen de commencer à utiliser LUIS rapidement.

Une fois le modèle conçu, testé et publié, il est prêt à recevoir et à analyser des énoncés via une requête http et répond avec des intentions extraites de la phrase de l'utilisateur et via les modèles créés ou prédéfinis. L'application envoie la phrase écrite par l'utilisateur à LUIS pour qu'il effectue une évaluation de la phrase et la renvoie à l'application au format JSON. L'application peut alors traiter les différents cas.

2.1 Les différents concepts de LUIS

2.1.1 Une intention :

L'intention dans LUIS représente les actions que l'utilisateur souhaite réaliser. L'intention est le sens ou le but que l'utilisateur veut exprimer avec sa phrase. Comme par exemple la prise d'un rendez-vous, la réservation d'une table pour un restaurant ou bien encore la recherche d'une liste de documents. Le développeur définit et nomme les intentions qui correspondent à ces actions. Pour la prise d'un rendez-vous l'intention aura par exemple le nom de *« rendez-vous »*

[1] Language Understanding Intelligent Service

2.1.2 Un énoncé :

L'énoncé dans LUIS est une entrée de texte de l'utilisateur que LUIS doit comprendre. Il peut s'agir d'une phrase, comme *« Prends un rendez-vous avec Nelson »,* ou d'un fragment de phrase, comme *« rendez-vous », « meeting avec Nelson »*. Les énoncés ne sont pas toujours bien formés, il peut y avoir plusieurs variations d'énoncés pour une intention en particulier.

2.1.3 Une entité :

L'entité dans LUIS représente des informations détaillées pertinentes dans l'énoncé *« Prendre un rendez-vous avec Nelson », « Nelson »* est un nom donc une entité. En reconnaissant et en étiquetant les entités mentionnées dans les énoncés, LUIS aide à choisir l'action spécifique à sélectionner pour répondre à la demande de l'utilisateur.

2.2 Accès à LUIS

Il y a deux façons de créer un modèle pour LUIS :

- Les API REST.
- Le site Web LUIS.

Les deux méthodes permettent de contrôler la définition de votre modèle LUIS. Utiliser le site Web LUIS ou les API de création, ou une combinaison des deux pour créer votre modèle. Cette gestion inclut des modèles, des versions, des collaborateurs, des API externes, des tests et des entraînements.

Une fois le modèle créé et publié, il faut passer l'énoncé à LUIS et recevoir les résultats de l'objet JSON avec les API REST.

2.3 Créez votre modèle LUIS

Commencez votre modèle LUIS avec des intentions que l'application client peut résoudre. Les intentions sont juste des noms tels que *« Rendez-vous »* ou *« Contact ».*

Une fois l'intention identifiée, vous avez besoin d'exemples d'énoncés que vous souhaitez que LUIS mette en correspondance avec votre intention, par exemple *« Prendre rendez-vous avec Nelson».* Ensuite, étiquetez les parties de l'énoncé qui sont pertinentes pour le domaine de votre application en tant qu'entités et définissez un type tel que la date ou le lieu.

Généralement, une intention est utilisée pour déclencher une action et une entité est utilisée comme paramètre pour exécuter une action.

Par exemple, une intention *« Rendez-vous »* pourrait déclencher un appel API vers un service externe pour réserver, prendre un rendez-vous dans un agenda, ce qui nécessite des entités telles que le nom de la personne, la date, l'heure, le lieu.

2.3.1 Identifier les entités

L'identification de l'entité détermine avec quel succès l'utilisateur final obtient la bonne réponse. LUIS fournit plusieurs façons d'identifier et de catégoriser les entités.

2.3.1.1 Entités prédéfinies :

LUIS possède de nombreux modèles de domaine prédéfinis, notamment des intentions, des énoncés et des entités préconstruites. Vous pouvez utiliser les entités prédéfinies sans avoir à utiliser les intentions et les énoncés du modèle prédéfini. Les entités préconstruites vous font gagner du temps.

2.3.1.2 Entités personnalisées :

LUIS vous offre plusieurs moyens d'identifier vos propres entités personnalisées, y compris les entités simples, les entités composites, les entités de liste, les entités d'expression régulière, les entités hiérarchiques et les entités de mots clés.

Complétement finalisé, il faut entrainer le modèle en cliquant sur le bouton Train. Cela permet à LUIS de comprendre le nouveau modèle pour qu'il puisse afficher le nombre d'énonciations associées à chaque intention et ainsi le pourcentage de la prédiction par énonciations.

Figure 1 : Barre de navigation de LUIS

Pour utiliser notre service, suivez les indications suivantes. Cliquez sur le bouton Test et écrivez votre phrase de test. On constate que pour les différentes phrases testées, il prédit systématiquement la bonne intention avec l'entité correspondante.

2.4 Comment améliorer les performances de LUIS

Une fois que votre application est publiée et que de véritables énoncés d'utilisateur sont entrés, LUIS fournit plusieurs méthodes pour améliorer la précision de la prédiction.

2.4.1 Apprentissage actif :

LUIS fournit des énoncés réels qu'il vous est relativement difficile de passer en revue. Vous pouvez les étiqueter en fonction des entités, les recycler et les republier. Ce processus itératif présente d'énormes avantages. Votre aide mène à l'amélioration maximale des performances du système. LUIS apprend plus vite grâce à cette méthode.

2.4.2 Listes de phrases :

Les listes de phrases LUIS fournit des listes de phrases pour vous permettre d'indiquer des mots ou des expressions significatives pour votre domaine d'application.

2.4.3 Patterns :

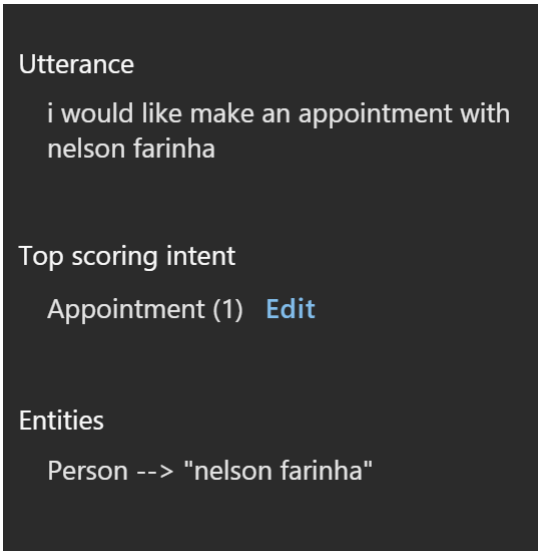
Les Patterns vous permettent de simplifier la collecte d'énoncés d'intention dans des modèles communs, comme le choix de mots et l'ordre des mots.

2.5 Configuration

Cas concret :

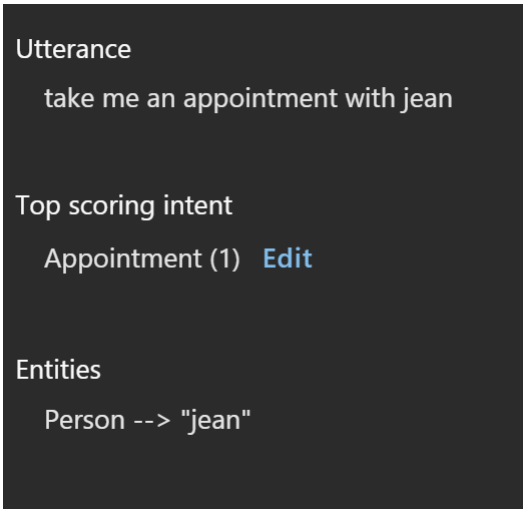
Si l'utilisateur demande à LUIS "*I would like make an appointment with Nelson Farinha".* L'intention de l'utilisateur est donc une prise de rendez-vous avec *« Nelson Farinha ».* L'intelligence du système porte sur sa capacité à qualifier la pertinence de la requête d'identifier l'action et le nom du protagoniste en question. C'est là où LUIS fait toute la différence car il est capable de déterminer deux choses dans cette phrase. L'intention de la phrase, qui est une prise de rendez-vous et la deuxième chose, la personne avec qui l'utilisateur veut prendre le rendez-vous dans LUIS, Cela s'appelle une entité. Dans l'exemple ci-dessous, on visualise que le service a bien reconnu le sens de l'intention et la personne avec qui il veut prendre un rendez-vous.

Figure 2 : Résultat de test de LUIS pour la phrase *« I would like make an appointment with nelson farinha »*



Nous relançons le moteur avec une autre question pour la prise de rendez-vous. On voit que LUIS a bien interprété automatiquement notre question.

Figure 3 : Résultat de test de LUIS pour la phrase *« take me an appointment with jean »*

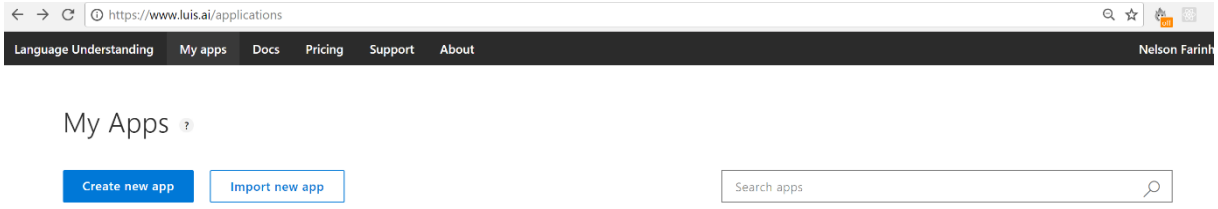


Le constat est la capacité de LUIS d'interpréter et de contextualiser les phrases des utilisateurs.

2.5.1 Introduction à l'interface de LUIS

Pour utiliser LUIS il faut se connecter à sa console en ligne www.LUIS.ai , puis créer votre nouvelle application.

Figure 4 : Page d'accueil pour la création d'une nouvelle application LUIS



LUIS intègre un ensemble de fonctionnalités qui permettront de nous fabriquer notre service parfait :

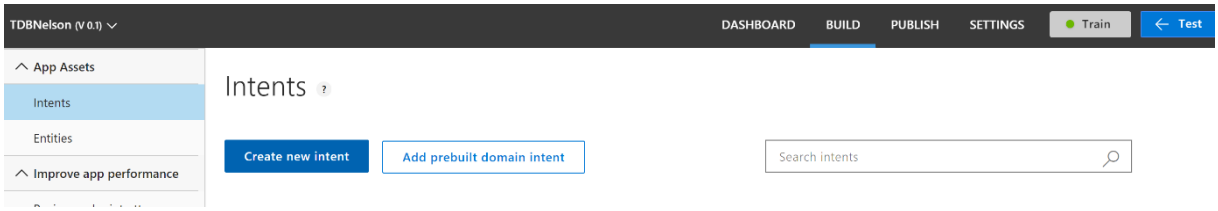
Intent : C'est l'intention de la phrase qu'on cherche à trouver.

Entities : Ce sont les entités, elles peuvent être personnalisées ou bien alors LUIS nous propose déjà un ensemble d'entités prédéfinies.

Utterances : Ce sont les différentes phrases que l'on fait apprendre à LUIS par intention.

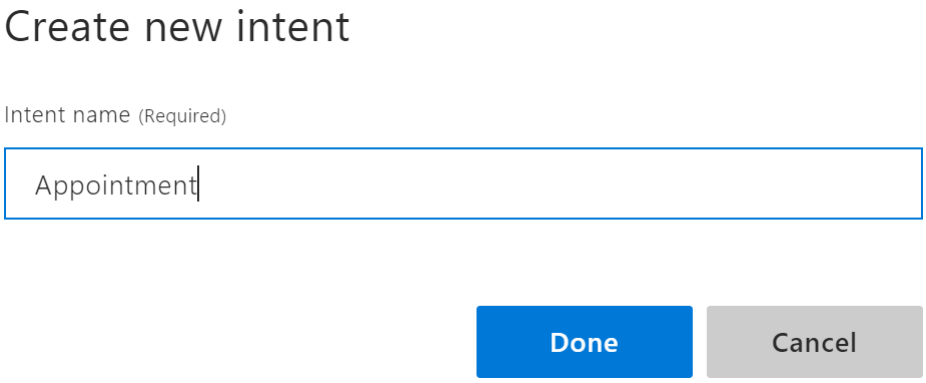
Pour créer une nouvelle intention allez dans l'onglet Intents puis cliquez sur Create new intent

Figure 5 : Page de création d'une nouvelle intention



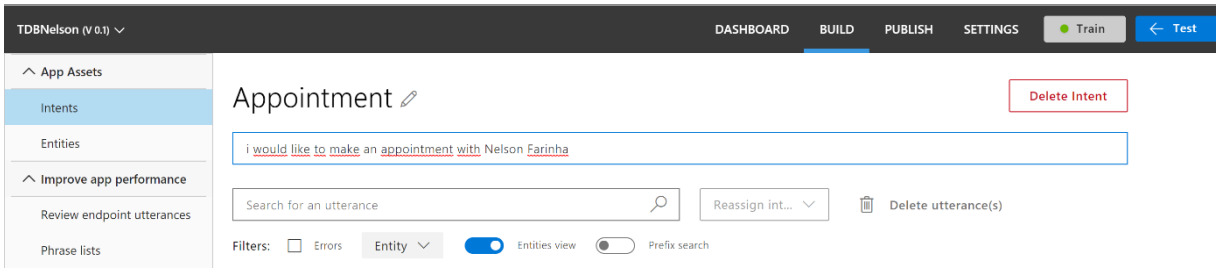
Donnez un nom à votre intention, pour l'exemple nous gardons notre cas de prise de rendez-vous.

Figure 6 : Zone de nommage d'une nouvelle intention



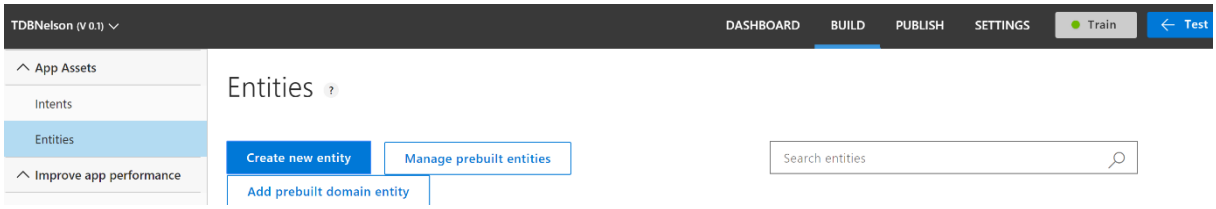
Définissez la phrase ou la suite de mots que LUIS devra interpréter.

Figure 7 : Page d'ajout des phrases clé pour une intention



On va ensuite créer notre première entité personnalisée. Pour cela il faut aller dans l'onglet Entities puis Creat new entity.

Figure 8 : Page de création d'une nouvelle entité



Il faut donner un nom à notre nouvelle entité, ici on l'appelle Personne. Puis un type qu'on peut choisir entre :

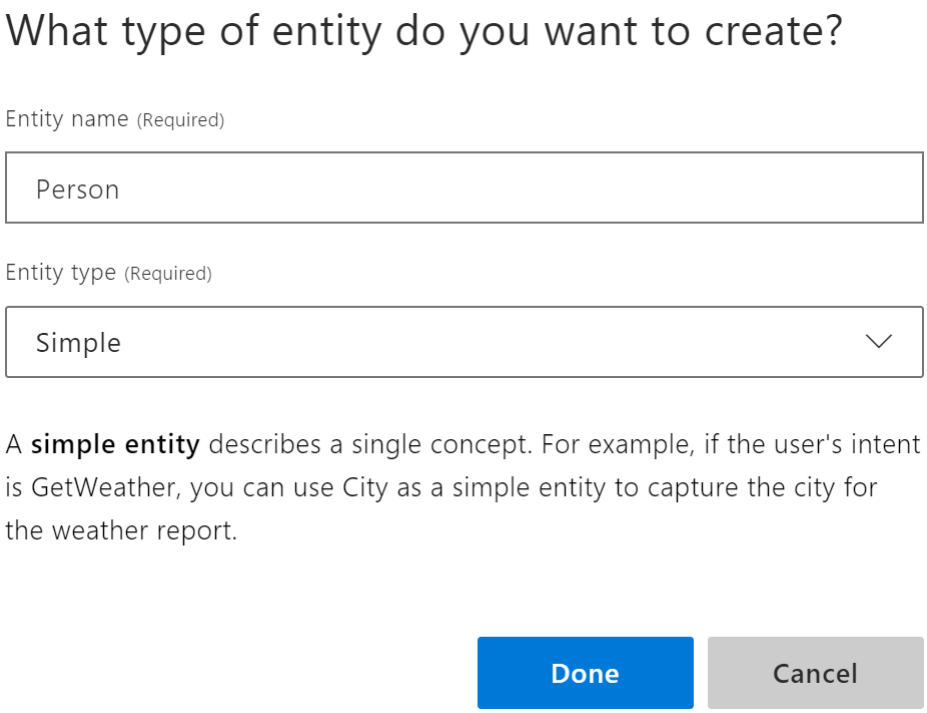
Simple : Une entité simple est une entité générique qui décrit un concept unique appris par LUIS.

Hierarchical : Est identique à une entité simple sauf qu'elle est hiérarchisée sous la forme d'une relation parent-enfant.

Composite : C'est un ensemble d'entités qui représente une entité composite.

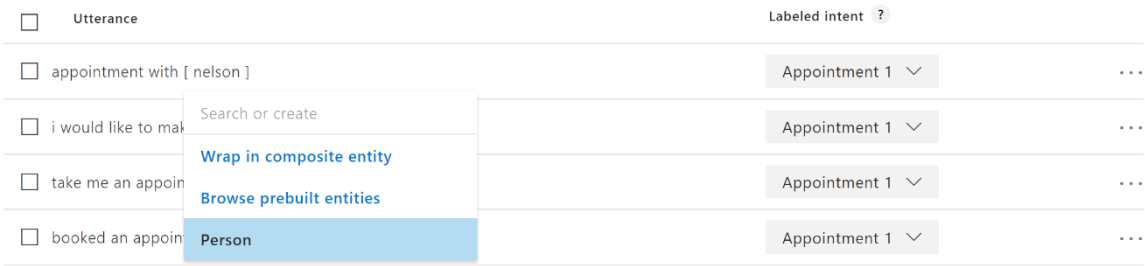
List : Les entités de liste représentent un ensemble fixe de valeurs.

Figure 9 : Zone de création d'une entité



Retournez dans la page qui représente votre intention puis sélectionnez l'élément qui correspond à votre entité.

Figure 10 : Modèle de LUIS avec ajout d'une entité

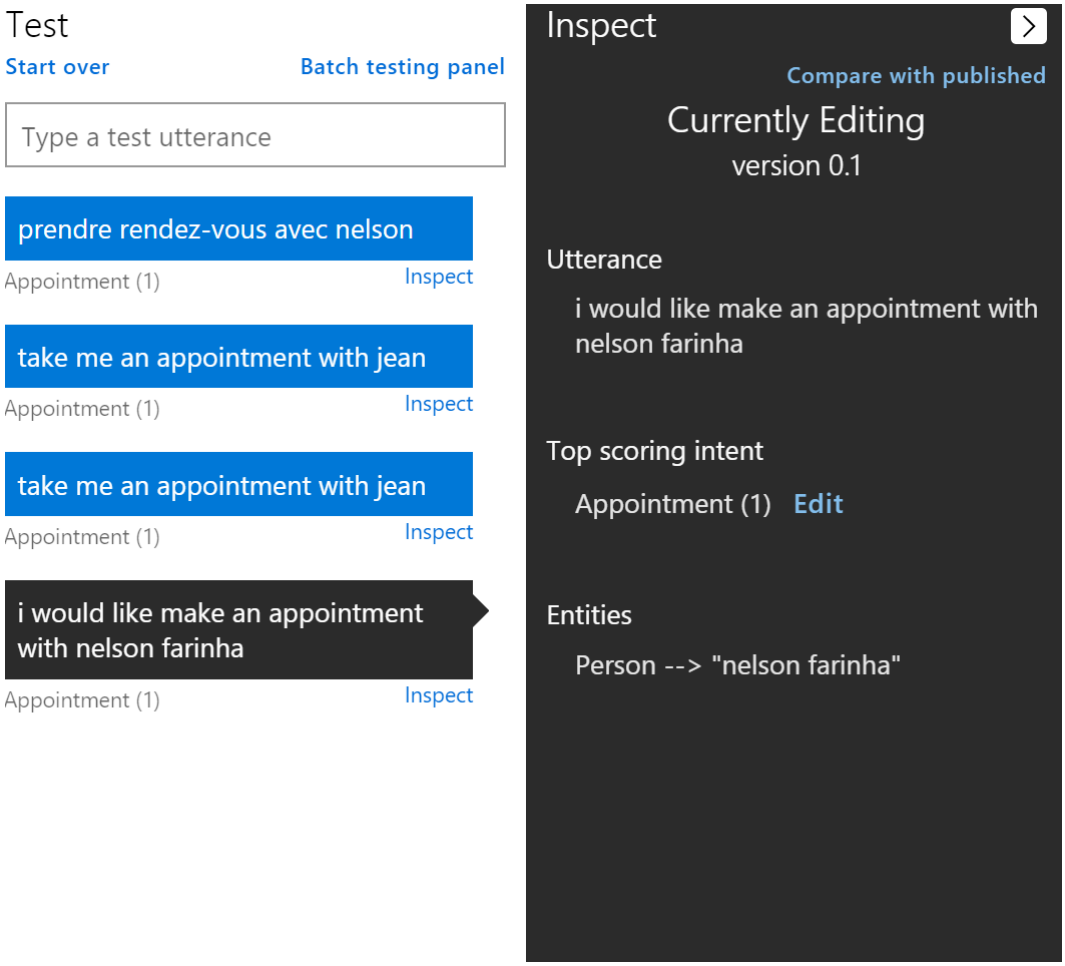


Une fois votre intention complètement finalisée, il faut entrainer le modèle en cliquant sur le bouton Train. Cela permet à LUIS de comprendre le nouveau modèle pour qu'il puisse afficher le nombre d'énonciations associées à chaque intention et ainsi le pourcentage de la prédiction par énonciation.

Figure 11 :

Nous pouvons maintenant tester notre service. Cliquez sur le bouton Test et écrivez votre phrase de test. On constate que pour les différentes phrases testées, il prédit systématiquement la bonne intention avec l'entité correspondante.

Figure 12 : Panneau de test de LUIS



2.6 Capacités de LUIS au travers d'un jeux de tests

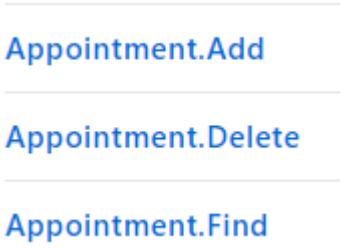
Pour comprendre la puissance de LUIS, étudions un cas complexe sur la prise de rendez-vous. Le but de ce cas est de voir les qualités, limites et techniques d'entrainement de LUIS. LUIS doit être capable de fournir toutes les informations nécessaires pour la prise de rendez-vous qui sont :

- L'action (créer un rendez-vous ou lister les rendez-vous ou supprimer un rendez-vous)
- La personne désirée pour le rendez-vous.
- La date du rendez-vous.
-

2.6.1 Construction des modèles :

Pour commencer, créons trois intentions pour pouvoir déterminer l'action que l'utilisateur souhaite exprimer.

Figure 13 : Intention de LUIS



Pour la personne, j'ai créé une entité Simple qu'il faudra entrainer à reconnaitre des personnes lors de la création du modèle. Pour la date, j'ai utilisé une entité qui est déjà toute implémentée dans LUIS : datetimeV2 qui va reconnaitre automatiquement à chaque fois qu'on parle de date.

Figure 14 : Model Appointment.Add

Figure 15 : Model Appointment.Delete

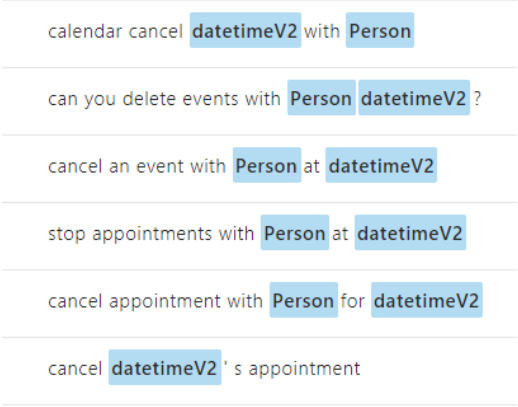
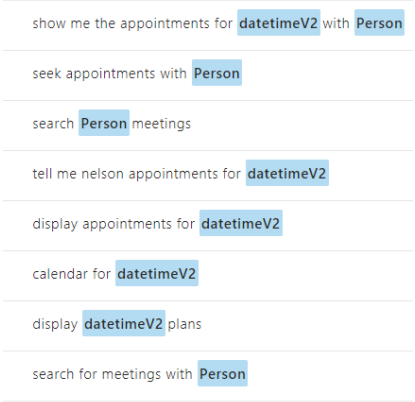


Figure 16 : Model Appointment.Find



2.6.2 Tests

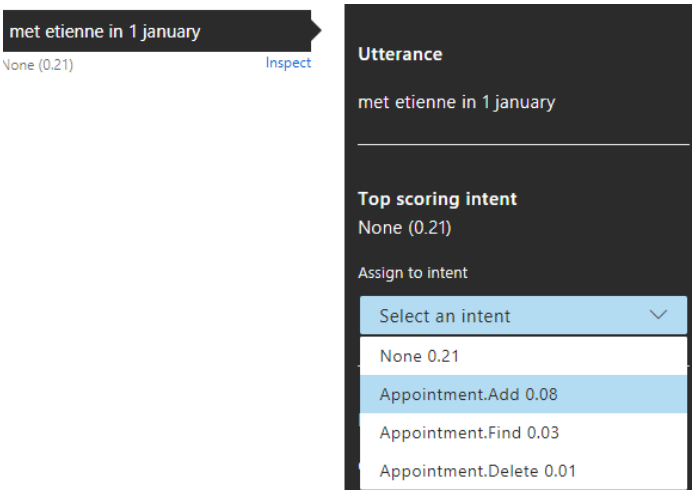
Tableau 1 : Tableau de test pour LUIS

	Action			Entities		Erreur
Add sentence	Add	Find	Delete	Person	Datetime	None
1 Make an appointment with Nelson	0.54			Nelson		
2 Make an appointment with Nelson for 24 March	0.76			Nelson	24 March	
3 Add a meeting for tomorrow with Bruno	0.78			Bruno	Tomorrow	
4 Schedule appointment for June 25 with Isabel Farinha	0.54			Isabel Farinha	June 25	
5 Met Etienne in 1 January				Etienne	1 January	0.21
6 Save the date May 17 with Jean Francois Piege	0.82			Jean Francois	May 17	
7 Can you make an appointment for tomorrow with Nelson	0.85			Nelson	Tomorrow	
8 Plan me one meeting with Jean for 8 December	0.74			Jean	8 December	
9 Add an appointment with Nelson for tomorrow	0.78			Nelson	Tomorrow	
Find sentence						
10 Telle me Nelson appointments for 25 June		0.94		Nelson	25 June	
11 Seek the appointments with nelson		0.68		Nelson		
12 Search for meetings with Bruno		0.98		Bruno		
13 Search Isabel meetings		0.95		Isabel		
14 Display appointments for 23 March		0.98			23 March	
15 Show me the appointments for tomorrow with Jean		0.96		Jean	Tomorrow	
Delet setence						
16 Cancel appointments for tomorrow with Nelson			0.95	Nelson	Tomorrow	
17 Can you delete events with Bruno tomorrow			0.91	Bruno	Tomorrow	
18 Cancel an event with Jean at 25 March			0.96	Jean	25 March	
19 Stop appointments with Isabel at 25 June			0.78	Isabel	25 June	
20 Cancel appointments with Etienne at 1 January			0.95	Etienne	1 January	
21 Cancel tomorrow's appointments			0.93		Tomorrow	

2.6.2.1 Conclusion test :

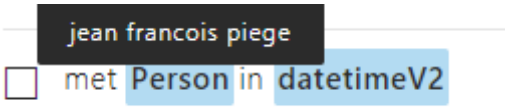
Après avoir déroulé le jeu de test on peut voir que pour le test 5 il n'a pas réussi à trouver l'intention, il faut donc entrainer LUIS pour cette question en lui sélectionnant la bonne intention.

Figure 17 : Exemple d'entrainement de LUIS



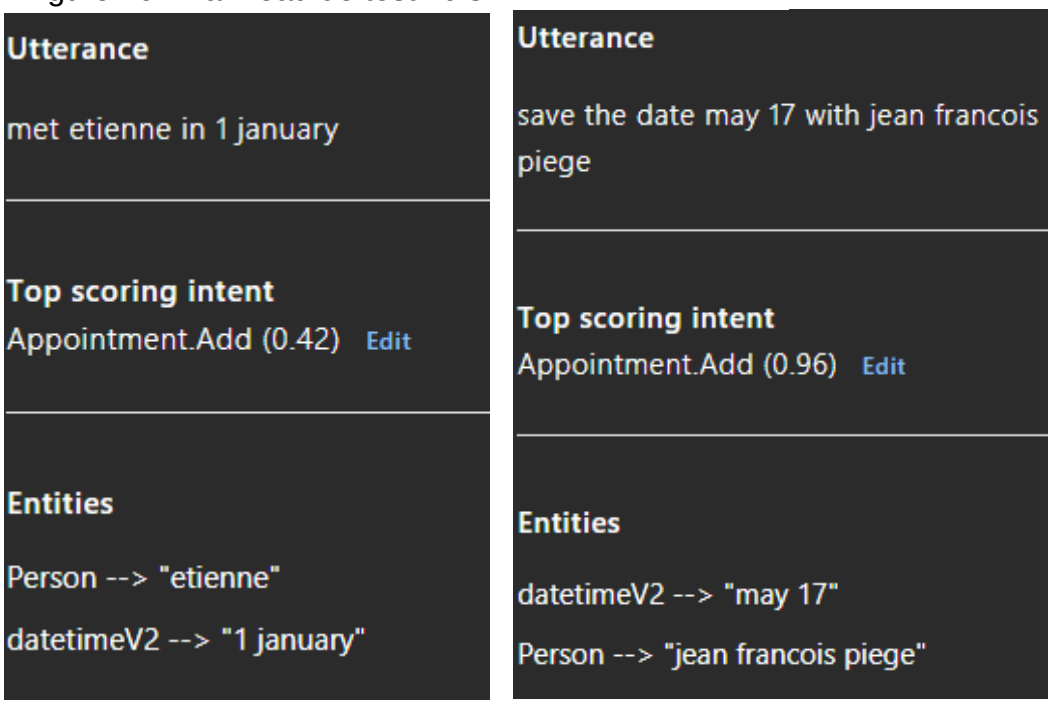
Pour les autres cas, LUIS n'as pas fait de grandes erreurs sauf pour la 6 ou il n'a pas détecté le nom en entier de «Jean Francois Piege ». Pour corriger ce petit défaut il faut lui renseigner un nom et un prénom composé dans un des énoncés d'une intention et l'étiqueter à l'entité Personne.

Figure 18 : Ajout d'une entité dans une phrase d'un modèle



Erreurs corrigées :

Figure 19 : Panneau de test Luis



2.7 Résumer

Nous pouvons constater que LUIS est capable d'analyser un cas complexe de compréhension de langage naturel, comme la prise de rendez-vous avec les différentes contraintes exprimées plus haut. Grâce aux différentes informations renvoyées on peut donc effectuer l'action que l'on souhaite dans notre application. Il suffit d'extraire l'intention qui représente l'action et les entités du code JSON retourné par LUIS. Un des points très positif, c'est que plus on utilise LUIS plus il devient performant et les pourcentages de ses réponses justes augmentent.

3. QnA Maker

QnA Maker est un service qui permet d'ajouter des questions qui sont liées à leur réponse. L'un des buts de notre bot est de pouvoir répondre à des questions, pour cela nous allons utiliser ce service. Dans de nombreux cas, les questions et réponses sont déjà présentes dans les FAQ. Grâce à QnA Maker, on peut absorber ses FAQ déjà existantes sous plusieurs formats qui sont :

- Txt
- Pdf
- Url
- Doc

Nous pouvons aussi écrire manuellement dans QnA Maker nos questions et réponses associées. Ce service permet de stocker ainsi plusieurs questions et réponses. Il utilise l'apprentissage automatique pour extraire des paires de questions et réponses qui ont du sens dans notre contenu. Il utilise également de puissants algorithmes de correspondance et de classement pour fournir la meilleure correspondance possible entre la requête de l'utilisateur et les questions.

3.1 Fonctionnement de QnA Maker

QnA Maker propose deux services clés pour le traitement de nos données :

L'Extraction : Les données structurées de questions-réponses sont directement extraites lors de la création du service, des sources de données semi-structurées comme les FAQ, les manuels produits.

La Correspondance : Une fois notre source de données formée et testé. Nous pouvons poser des questions à QnA maker qui lui va déterminer la meilleure réponse par rapport à l'entrainement réalisé.

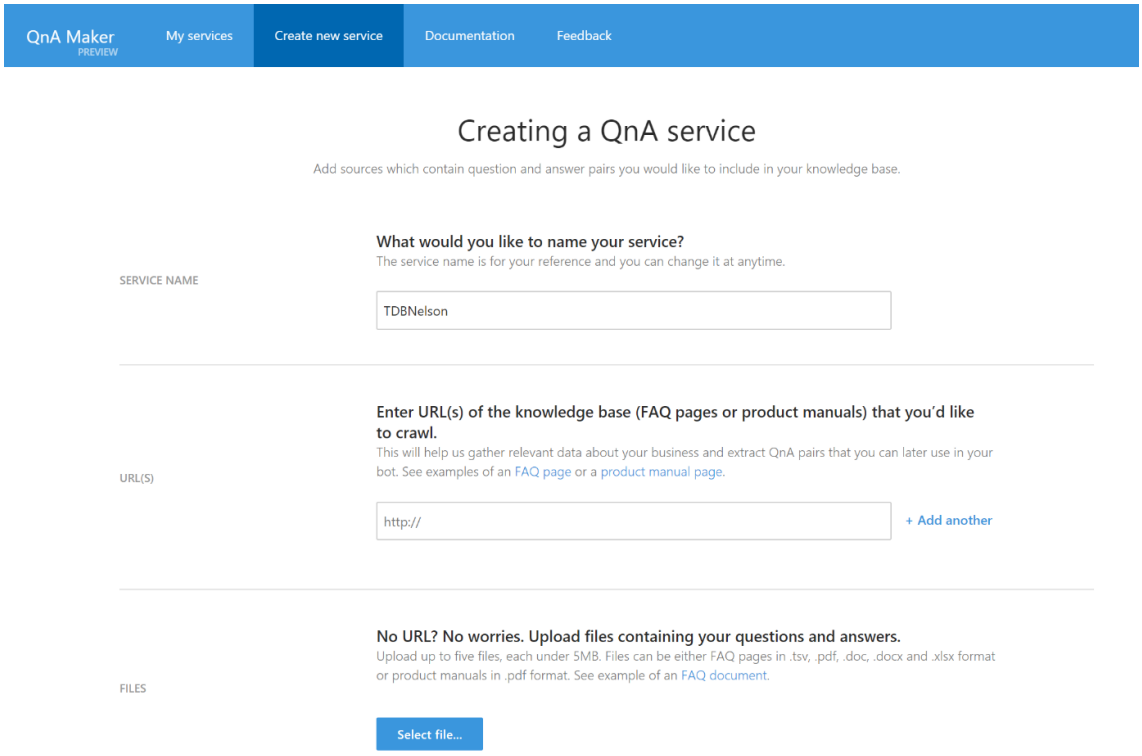
3.2 Configuration de QnA Maker

QnA Maker propose une interface graphique très simplifiée nous permettant d'ajouter des nouveaux FAQ, de les tester et les publier.

Pour créer votre premier service QnA allez sur le site https://qnamaker.ai. Cliquer sur l'onglet Create new service. Il faut donner un nom à notre service et sélectionner notre

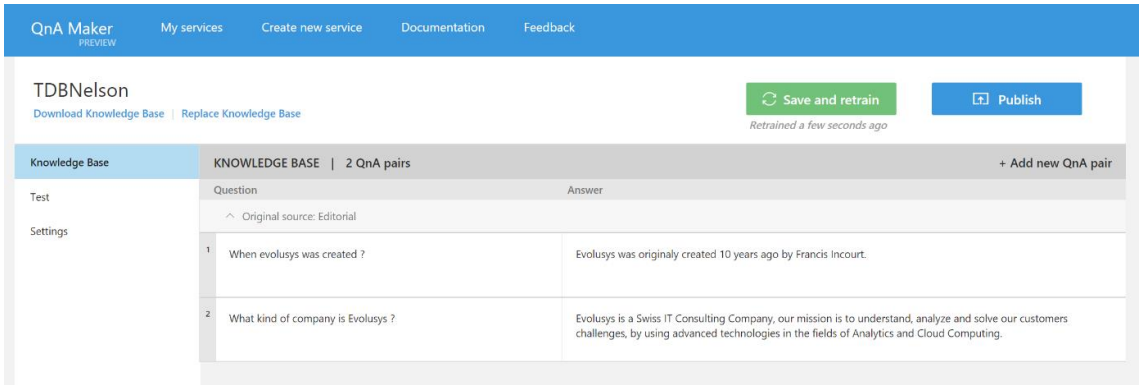
FAQ par un des formats évoqué plus haut ou sinon le créer à la main dans l'interface graphique.

Figure 20 : Page d'accueil de création d'un nouveau service de QnA Maker



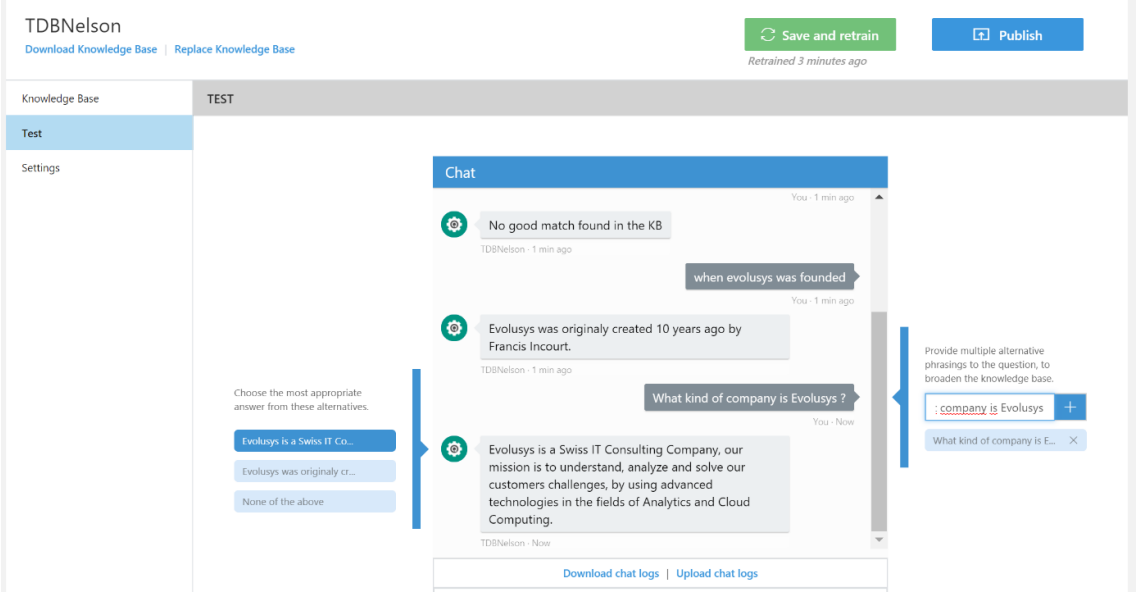
Une fois dans l'onglet Knowledge Base écrivez vos FAQ puis appuyez sur Save and retrain.

Figure 21 : Page de nos question réponses



Pour tester notre FAQ, nous allons poser des questions dans l'interface graphique de test. Dans la zone d'insertion de texte à droite de notre chat, nous pouvons écrire différentes alternatives à notre question. Si notre service répond mal à une question que l'on lui pose nous pouvons choisir la bonne réponse manuellement à gauche du chat. Une fois vos tests réalisés, cliquer sur Save and retrain pour que le service puisse apprendre des actions effectuer lors des tests.

Figure 22 : Page de test de QnA Maker



Une fois vos tests réalisés il faut publier l'entrainement, pour cela, cliquez sur Publish.

Figure 23 : Page de publication de QnA Maker

3.3 Résumer

Grace à un outil simple comme QnA maker nous n'avons pas besoin d'utiliser de base de données externe pour stocker les réponses à nos questions. Ce service le fait pour nous. Un de ces grands avantage est qu'il dispose d'une interface simple d'utilisation, il n'y a pas besoin d'avoir de grandes compétences techniques pour le configurer. Nous pouvons aussi l'entrainer avec différentes questions, ce qui rend pertinent son intégration dans un bot intelligent.

4. Fiche technique Ms Translator

Microsoft Translator Text Api est un service cognitif de Microsoft. Ce service de traduction automatique fondé sur le cloud prend en charge plus de 60 langues. On peut facilement l'intégrer dans nos applications via des appels REST.

4.1 Fonctionnalités de Translator

Natif Neral : Pour les langues supportées la traduction automatique neuronal est utilisée par défaut. Cette Technologie de traduction automatique capture le contexte des phrases avant de les traduire dans la langue désirée. Ce qui donne un côté plus humain à la phrase traduite.

Détection : L'Api est capable de détecter automatiquement la langue du texte qu'elle reçoit en entrée.

Translittération : Elle est capable de convertir la phonétique en mot ou vice versa. Exemple : on veut convertir des caractères de l'alphabet mandarin en phonétique latine pour savoir comment les prononcer.

Dictionnaire bilingue : Propose plusieurs mots, des phrases en contexte avec la phrase à traduire pour nous aider à choisir la traduction optimale.

4.2 Résumer

Ms Translator fait plus qu'une simple traduction. Il fait une traduction avec des phrases avec un côté plus humaine. De plus cette Api est capable de détecter automatiquement la langue du texte qu'il reçoit en entrée.

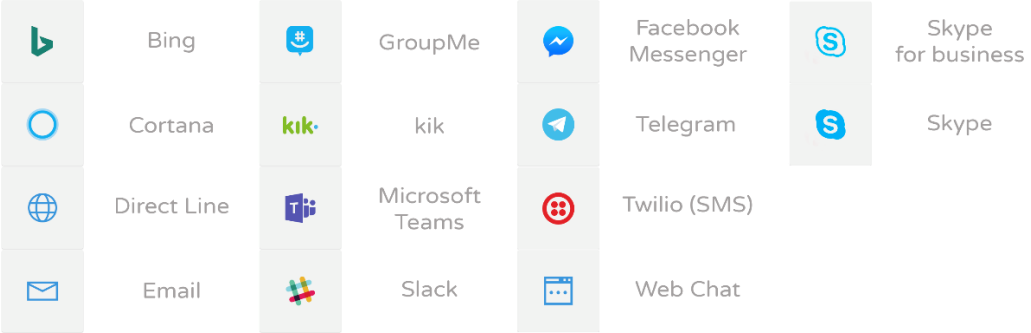
5. Bot Framework

Bot Framework est un produit de Microsoft qui permet de créer un environnement de développement pour les chatbot. Ce Framework fournit une palette d'outils pour tester, déployer et configurer des chats bot. Ce Framework est disponible dans deux langages de programmation, en C# et en Node.js. Il permet aux développeurs de créer des bots qui résolvent des problèmes métier.

Les outils utilisés pour notre chatbot :

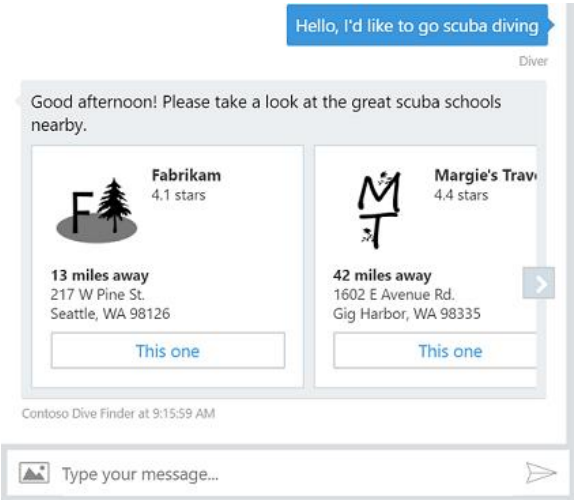
Les canaux de communication : Nous pouvons communiquer avec notre bot à travers différents canaux de communication listés ci-dessous.

Figure 24 : Canaux compatible avec Bot Framework



Les cartes adaptives : Permettent de présenter du contenu sous forme graphique que nous pouvons personnaliser et automatiquement adapter à nos différents canaux.

Figure 25 : Chat avec cartes adaptives personnalisé

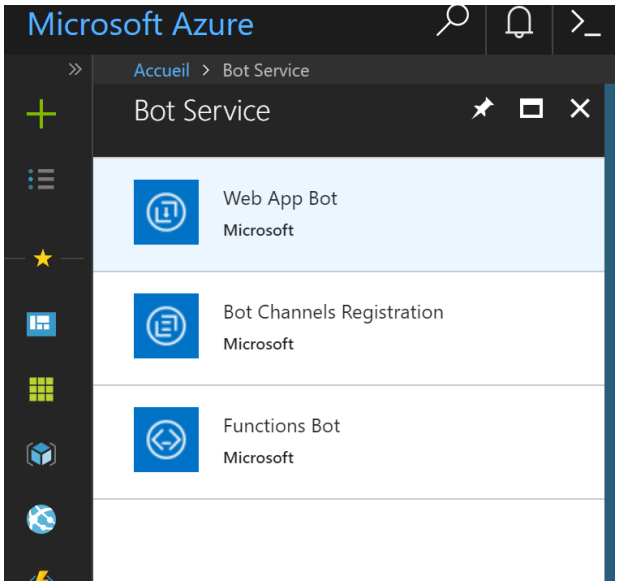


(https://docs.microsoft.com/en-us/adaptive-cards/get-started/bots)

5.1 Installation de l'environnement

Pour la création de son bot, il faut se rendre sur le site web https://dev.botframework.com et disposer d'un compte Azure. Cliquez sur Create a bot. Cette action va ouvrir votre console azure, dans la console choisissez Web App Bot.

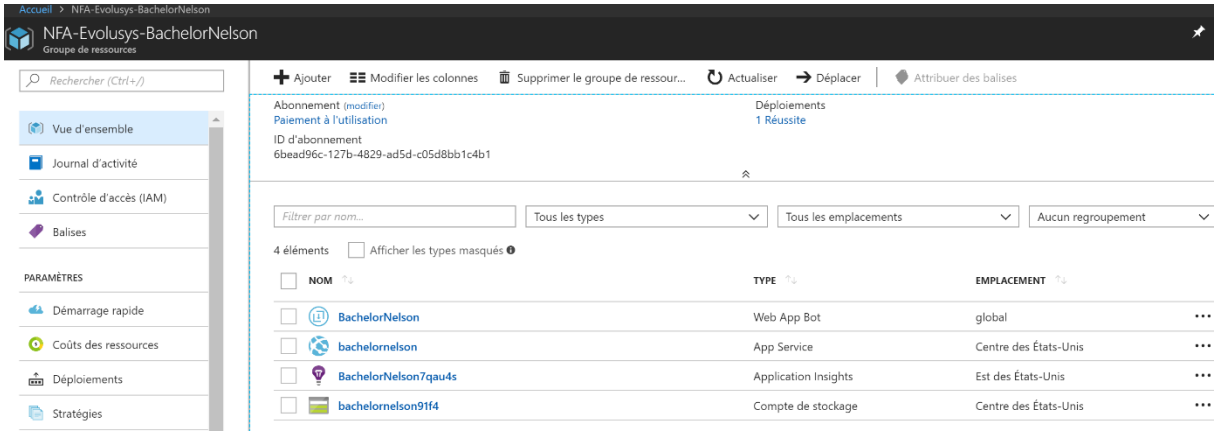
Figure 26 : Microsoft Azure création d'un projet chatbot



Voici le contenu du groupe de ressource de votre nouveau chatbot Il contient :

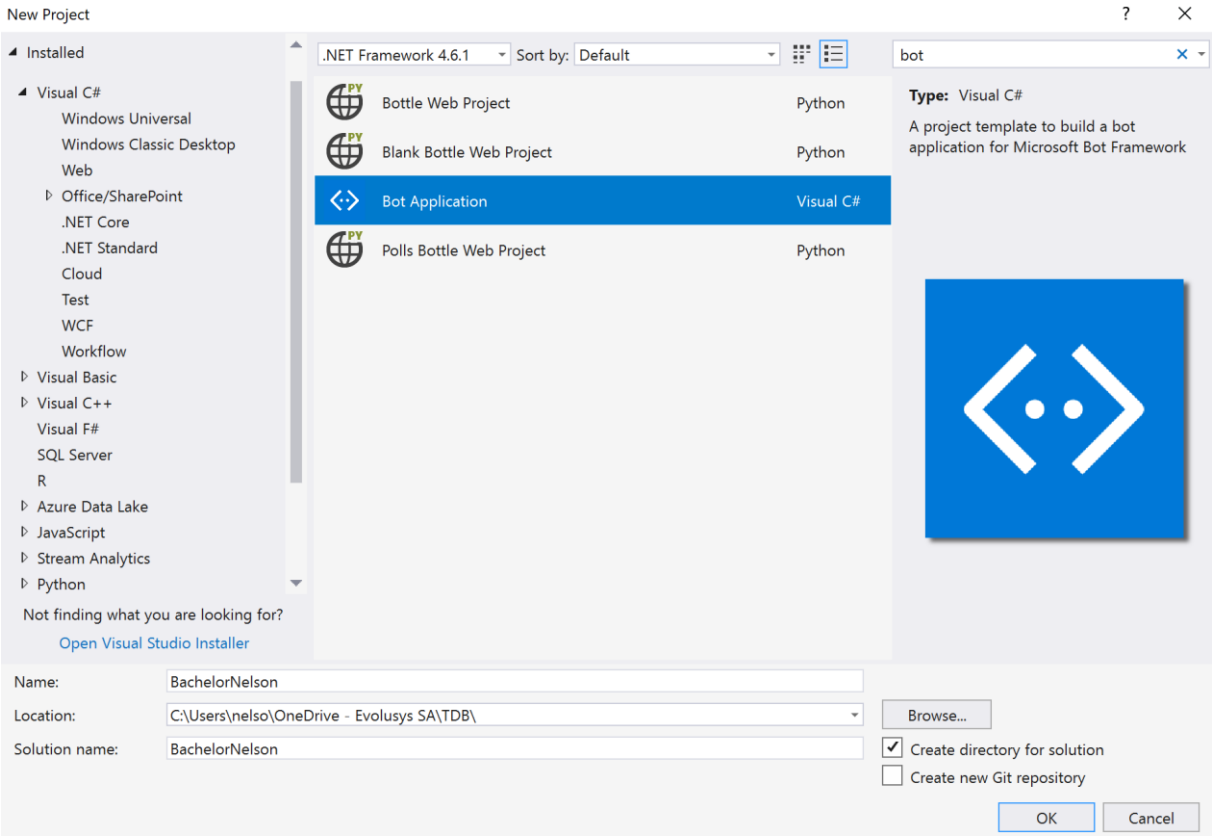
- **Une Web App Bot :** qui nous permettra de connecter le canal que l'on désire.
- **Une App service :** c'est là qu'on publiera notre code c#.
- **Une Application Insights :** pour avoir un visuel graphique avec les interactions avec le bot.
- **Un compte de stockage :** pour stocker des données.

Figure 27 : Élément d'un projet chatbot sur Azure



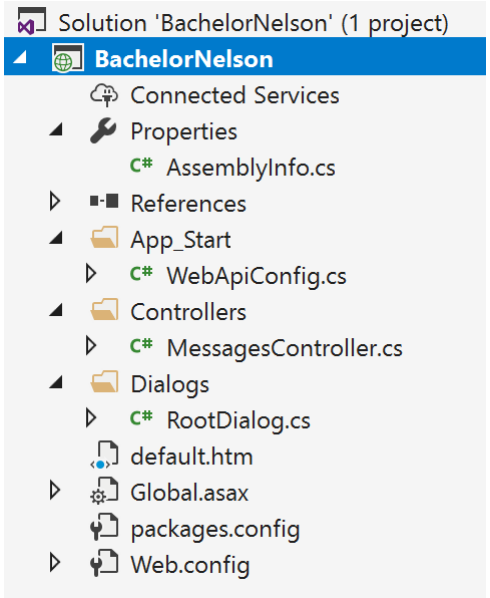
Maintenant nous allons utiliser Bot Application dans Visual studio pour créer notre bot. Pour cela, créer un nouveau projet dans Visual studio et sélectionner Bot Applicaton.

Figure 28 : Sélection du type de projet Bot Application



Voici le contenu de notre projet.

Figure 29 : Contenu du projet chatbot après sa création



Nous allons nous intéresser ici aux fichiers suivants :

- MessagesController.cs qui permettra de faire l'aiguillage des interactions avec notre bot.

- RootDialog.cs qui contient le traitement des messages reçus de l'utilisateur ainsi que le traitement de nos réponses.
- Web.congi qui contient nos différentes clés et url privée pour l'interaction avec les API en format XML

En regardant le code de MessagesController.cs on constate que notre bot est une simple web api avec une méthode POST qui sera appelée par nos channels.

Figure 30 : Code de la classe MessagesController.cs

```
/// <summary>
/// POST: api/Messages
/// Receive a message from a user and reply to it
/// </summary>
0 references
public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
{
    if (activity.Type == ActivityTypes.Message)
    {
        await Conversation.SendAsync(activity, () => new Dialogs.RootDialog());
    }
    else
    {
        HandleSystemMessage(activity);
    }
    var response = Request.CreateResponse(HttpStatusCode.OK);
    return response;
```

Dans la classe RootDialog.cs c'est là où nous allons interagir avec l'utilisateur, plus précisément dans la procédure MessageReceivedAsync où nous recevons un résultat exprimé par la signature de méthode result de type IAwaitable<object>. Avec laquelle nous arrivons à extraire le texte avec le code ci-dessous.

Figure 31 : Procédure MessageReceivedAsync de la classe RootDialog.cs

```
2 references | Nelson Farinha, 92 days ago | 1 author, 22 changes
private async Task MessageReceivedAsync(IDialogContext context, IAwaitable<object> result)
{
        var activity = await result as Activity;
        //The text of user
        string text = activity.Text;
```

Nous allons aussi interagir avec la signature de méthode Context de type IDialogContext qui permet d'envoyer du contenu à notre chat (text, cartes adaptives, graphiques, image, fichier). Avec le code suivant « await context.PostAsync(Le Contenu à envoyer) ; ».

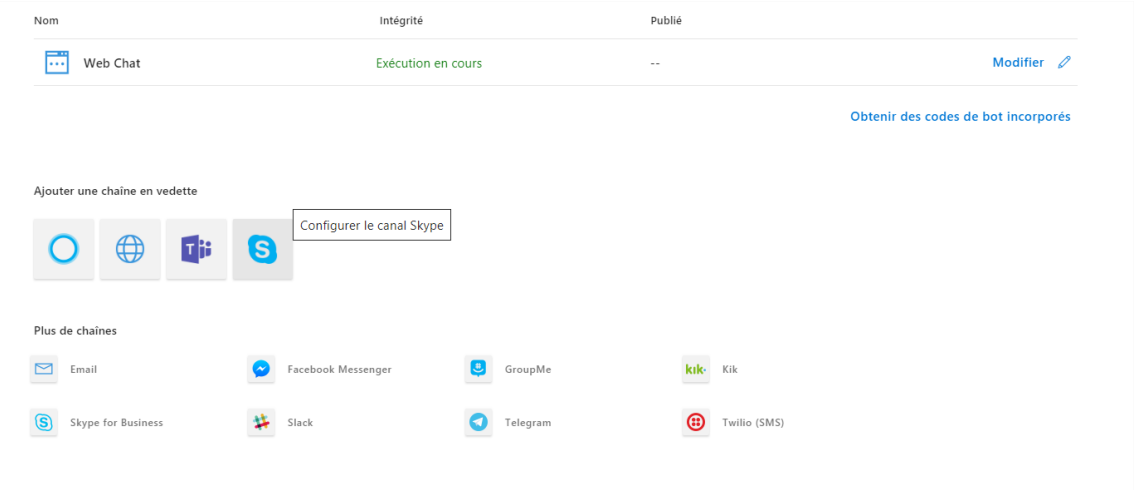
Figure 32 : classe Web.config avec les identifiants masqués

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=301879
  -->
<configuration>
  <appSettings>
    <!-- update these with your BotId, Microsoft App Id and your Microsoft App Password-->
    <add key="BotId" value="YourBotId" />
    <add key="MicrosoftAppId" value="" />
    <add key="MicrosoftAppPassword" value="" />
    <!--<add key="MicrosoftAppId" value="                                    "/>
    <add key="MicrosoftAppPassword" value="                        "/>-->
    <add key="CryptoProfileUrl" value="
    <add key="CryptoProfileKey" value="
    <add key="CryptoUrl" value="                                    " />
    <add key="LuisKey" value="                                  " />
    <add key="LuisID" value="                                    " />
    <add key="LuisURI" value="                                                                    " />
    <add key="knowledgebaseIdQnAMaker" value="                                        " />
    <add key="qnamakerSubscriptionKeyQnAMaker" value="                                    " />
    <add key="SubscriptionKey" value="                                " />
    <add key="CognitiveServicesTokenUri" value="
    <add key="TranslatorUri" value="                                                    " />
    <add key="StorageConnectionString" value="
  </appSettings>
```

5.2 Résumer

Grâce à ce Framework proposé par Microsoft nous avons déjà une base pour commencer à développer et ne partons pas de zéro. De plus, tous les outils sont directement dans le cloud, ce qui permet d'utiliser le bot n'importe où. L'un des plus gros avantages est l'automatisation des canaux de communication puisqu'il suffit de cliquer sur le canal souhaité pour que la création soit faite sans avoir à toucher à notre code.

Figure 33 : Page d'ajout de canaux de communication pour notre chatbot dans azure



Il en est de même pour les cartes adaptives. Nous les créons une seule fois et elles s'adaptent automatiquement aux différents canaux de communication.

6. Architecture

Pour que notre chatbot devienne intelligent et polyglotte nous devons le connecter à différents services cognitifs. Pour cela nous avons choisi l'architecture suivante avec les services cognitifs suivants :

LUIS : Qui permet d'interpréter une question sur l'entreprise Evolusys est de retourner l'intention de cette question.

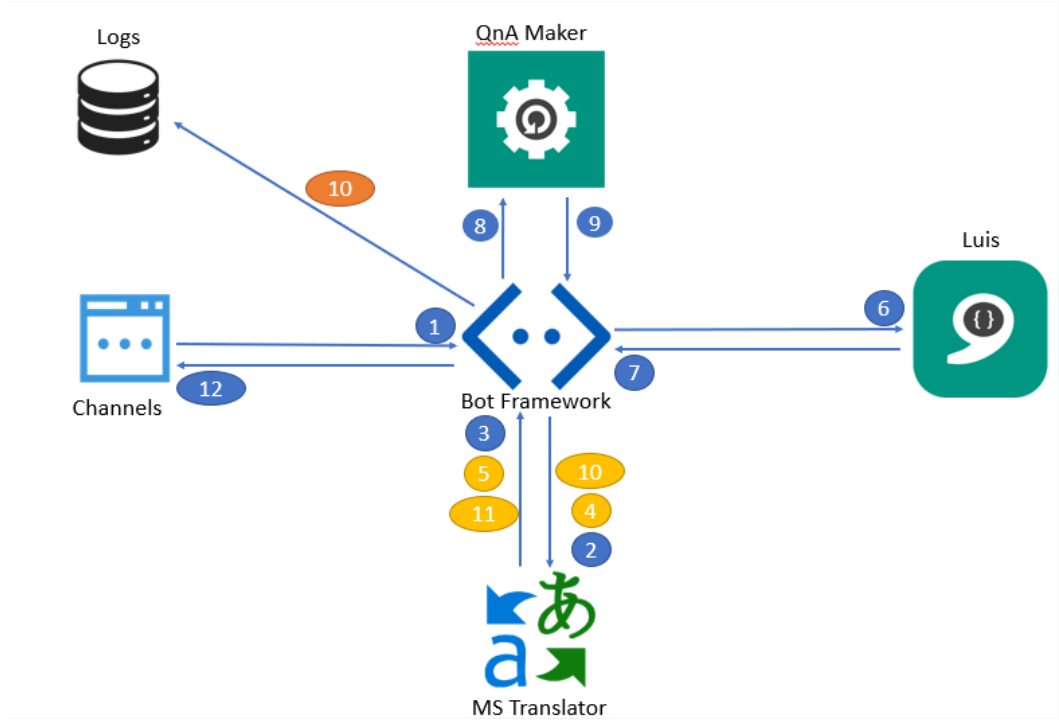
MS Translator : Qui permet de détecter la langue de la question posée par l'utilisateur et ainsi de pouvoir traduire la question et la réponse de l'utilisateur.

QnA Maker : Qui est un service de question réponse de Microsoft, qu'on utilise pour stocker nos questions réponses.

Nous allons traiter aussi les cas où notre bot ne comprend pas la question c'est-à-dire : Quand le score de l'intention de LUIS est inférieur à 65% ou que LUIS renvoie une intention nommée « None » ou encore Quand QnA Maker n'a pas la réponse à la question. Quand ces cas arrivent, nous allons stocker la question correspondante dans une table Logs pour pouvoir traiter ces questions et leur assigner une réponse dans le futur.

6.1 Explication de l'architecture

Figure 34 : Architecture du chatbot



1. L'utilisateur envoie une question au bot.

2. Le bot envoie la question à MS Translator.

3. Ms Translator envoie le code de la langue de la question au bot.

Si la question est dans une autre langue que l'anglais :

    4. Le bot là renvoie à MS translator pour la traduire en anglais.

    5. MS translator traduit la question en anglais et l'envoie au bot.

6. Le bot envoie la question en anglais à LUIS.

7. LUIS envoie l'intention de la question au bot.

8. Le bot envoie la question en anglais à QnA Maker.

9. QnA Maker envoie une réponse à la question au bot.

Si la Question n'a pas été comprise ni par LUIS ni par QnA Maker

    10. Stocke la question dans les logs.

Si la langue de la question est différente de l'anglais :

    10. Le bot envoie la réponse à Ms Translator pour la traduire.

    11. Ms Translator envoie la réponse traduite au bot.

12. Le bot envoie la réponse à l'utilisateur.

6.2 Résumer

Nous avons estimé que cette architecture était adaptée à l'évolution de cette solution dans le futur pour lui brancher plus tard d'autres services cognitifs ou lui faire apprendre encore plus de contenu. L'avantage d'utiliser ces services Microsoft est que nous sommes dans un environnement où il n'y a pas de problème de compatibilité avec les différents services.

7. Analyse du chatbot

Pour réaliser l'analyse de ce chatbot nous avons imaginé un jeu de test sous forme de questions que nous allons dérouler. Ainsi nous pourrons constater que les réponses renvoyées par le bot correspondent bien à nos attentes.

7.1 Jeu de tests

Tableau 2 : Jeu de test final

N°	Questions	Réponses
1	I would like make an appointment	OK
2	Prendre rendez-vous	OK
3	What types of business cases Evolusys offers?	OK
4	Quel genre de cas d'affaires proposez-vous ?	OK
5	Montre-moi l'équipe de direction ?	OK
6	What is the managment team?	OK
7	Where are you based?	OK
8	Quelle est l'adresse d'Evolusys ?	OK
9	When are your next event?	OK
10	Quand aura lieu votre prochain évènement ?	OK
11	What kind of services Evolusys do?	OK
12	Quels types de services proposez-vous ?	OK
13	When evolusys was created ?	OK
14	What is evolusys ?	OK

15	Combien de personnes travaillent pour Evolusys ?	OK
16	What about Advanced Analytics applied to People Segmentation?	Le bot ne peut pas encore répondre cette question et la stocke dans les logs
17	Who are Etienne Marie?	Le bot ne peut pas encore répondre cette question et la stocke dans les logs
18	Peux-tu me montrer toute l'équipe d'Evolusys ?	Le bot ne peut pas encore répondre cette question et la stocke dans les logs
19	Puis-je avoir plus d'informations sur le service des solutions ?	Le bot ne peut pas encore répondre cette question et la stocke dans les logs

7.1.1 Explication des tests

Nous constatons que le bot renvoie les réponses attendues quand on a entrainé LUIS à identifier ces questions, ou bien quand on a rempli QnA maker avec ces questions. Quand les questions ne sont pas assez pertinentes pour ces services, le bot répond « Sorry I can't answer this yet » et stocke la question dans les logs pour un traitement futur et un apprentissage continu du bot.

8. Conclusion

A travers ce mémoire, nous avons appris à utiliser et configurer quelques services cognitifs de Microsoft. Nous avons aussi pu observer une architecture de chatbot lié à des services cognitifs et qui est évolutive.

Nous pouvons imaginer qu'à l'avenir nous serons capables de parler oralement à notre bot dans n'importe quelle langue. Qu'il nous répond dans la langue en question grâce à notamment l'API Speech de Microsoft. Ou bien encore que notre bot sera capable d'analyser le sentiment des questions de l'utilisateur avec l'Api Text Analytics de Microsoft. Il y a mille et une manières de rendre notre bot encore plus humain avec l'intelligence artificielle.

Nous sommes encore loin d'avoir fini ce projet chez Evolusys, mais nous partons sur une base très solide. Donna, le nom qu'on a donné à ce chatbot va bénéficier de nombreuses améliorations à l'avenir. Nous avons déjà pu présenter Donna à nos clients, beaucoup de ces clients sont déjà intéressés pour l'accueillir dans leur entreprise. Que ce soit comme un web assistant ou encore comme un service de support.

Figure 35 : Logo de Donna

En ce qui concerne l'avenir des chats bot je pense qu'ils vont faciliter le travail de nombreux collaborateurs, car ils seront de plus en plus « intelligents » et « humains », à un tel point que l'utilisateur ne fera plus la différence.

Sitographie

MICROSOFT, Documentation Microsoft de LUIS : https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/Home

MICROSOFT, Documentation Microsoft de QnA Maker : https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/overview/overview

MICROSOFT, Documentation Microsoft de Ms Translator : https://www.microsoft.com/en-us/translator/translatorapi.aspx

MICROSOFT, Documentation Microsoft de Framework bot : https://docs.microsoft.com/en-us/azure/bot-service/?view=azure-bot-service-3.0

INFINITE BLOGS, Thomas LEBRUN, 2016. Présentation de LUIS, ou comment rendre vos bots plus intelligents ! : https://blogs.infinitesquare.com/posts/divers/presentation-de-LUIS-ou-comment-rendre-vos-bots-plus-intelligents

INFINITE BLOGS, Thomas LEBRUN, 2016. Présentation et introduction au Bot Framework : https://blogs.infinitesquare.com/posts/divers/presentation-de-LUIS-ou-comment-rendre-vos-bots-plus-intelligents

MEDIUM, Rachhek Shrestha, 2017 What is LUIS.AI ? : https://medium.com/ai-for-developers/what-is-LUIS-ai-8ef7f972b7f7

DZONE, Murail K, 2018. An introduction to LUIS(Language Understanding Intelligent Service) : https://dzone.com/articles/LUIS-language-understanding-intelligent-service

Annexe 1 : Analyse des risques

Risques liés à LUIS

RI_01 : Mauvaise interprétation de l'intention de la phrase

Niveau de risque
Moyen
Description
LUIS interprète mal la phrase de l'utilisateur.
Impact
La réponse donnée par le bot ne sera pas la bonne, car l'intention ne sera pas la bonne.
Détection
Le bot nous renvoie une réponse hors sujet.
Mesure de prévention
Faire des tests et entrainer LUIS.
Plan B
Stocker les phrases mal interprétées et les faire apprendre à LUIS

Risques liés à Ms Translator

RI_02 : Mauvaise Traduction de la phrase de l'utilisateur

Niveau de risque
Haut
Description
La traduction réalisée par l'API Ms translator est incorrecte.
Impact
La réponse donnée par le bot ne sera pas la bonne, car LUIS ne comprendra pas l'intention.
Détection
Le bot nous renvoie une réponse hors sujet.
Mesure de prévention
Faire des tests sur Ms translator.
Plan B
Changer d'API de traduction.

RI_03 : Mauvaise Traduction de la phrase du bot

Niveau de risque
Haut
Description
La traduction réalisée par l'API Ms translator est incorrecte.
Impact
La réponse donnée par le bot sera incompréhensible.
Détection
Le bot nous renvoie une réponse hors sujet.
Mesure de prévention
Faire des tests sur Ms translator avec les réponses du bot.
Plan B
Changer d'API de traduction

RI_04 : Mauvaise détection de la langue de l'utilisateur

Niveau de risque
Haut
Description
La détection de la langue est fausse.
Impact
La réponse donnée par le bot sera dans une autre langue.
Détection
Le bot nous renvoie une réponse dans une autre langue que la nôtre.
Mesure de prévention
Faire des tests sur Ms translator avec différentes questions.
Plan B
Changer d'API de traduction

Risques liés à Risques liés à QnA Maker

RI_05 : Mauvaise réponse renvoyer par QnA Maker

Niveau de risque
Haut
Description
La réponse envoyée par QnA n'est pas la bonne
Impact
La réponse donnée par le bot ne sera pas bonne.
Détection
Le bot nous renvoie une réponse hors sujet.
Mesure de prévention
Faire des tests sur QnA maker et l'entrainer.
Plan B
Faire réapprendre la question avec la bonne réponse à QnA.

Autres risques

RI_06 : Le bot ne connait pas de réponse à la question.

Niveau de risque
Moyen
Description
Le bot ne connait pas la réponse à la question.
Impact
Le bot renvoie une erreur et est donc incompréhensible.
Détection
Le bot nous renvoie une erreur ou plante
Mesure de prévention
Gérer les exceptions dans le code.
Plan B
Gérer l'erreur et dire que le bot ne peut pas répondre pour le moment à cette question et enregistrer la question pour la traiter plus tard.