

ENSIIE  
Stage de première année

# État de l'art

## Conception et développement d'un agent conversationnel

Hugo BELHOMME

Jacky CASAS  
Omar ABOU KHALED

4 juillet 2017



# Sommaire

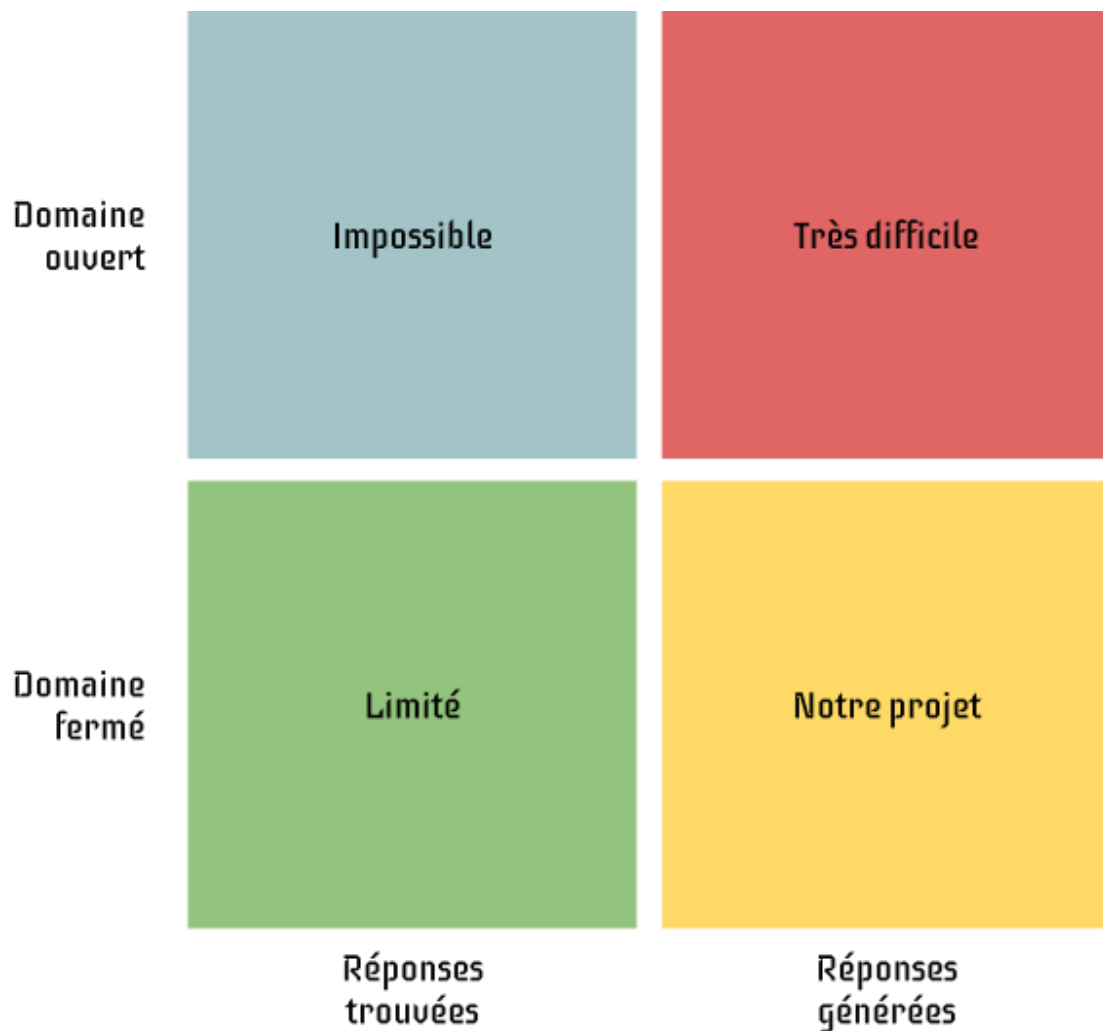
I. Qu'est-ce qu'un chatbot ?.....	3
II. Principe général de fonctionnement.....	4
III. Blocages classiques et solutions.....	5
IV. Comparaison des plateformes.....	6
V. Comparaison des services.....	7
VI. Microsoft bot framework.....	8

# Qu'est-ce qu'un chatbot ?

Un chatbot -ou agent conversationnel- est un programme informatique capable de répondre lorsqu'un utilisateur lui transmet un message. Ils sont utilisés dans le domaine de la communication, de la relation client, de l'aide à la personne ou des loisirs. Ainsi il existe des chatbots qui permettent de répondre à des questions simples de clients, qui peuvent prendre des rendez-vous ou passer des commandes pour l'utilisateur, qui peuvent répondre aux questions de l'utilisateur sur la santé ou qui peuvent faire jouer l'utilisateur à des quizz.

L'intérêt d'un chatbot est qu'il peut répondre correctement et instantanément à tout type de question pour laquelle il a été préparé. Ceci avec très peu de coûts d'entretien par rapport à des employés puisque la majeure partie du travail est effectuée en amont.

Il existe de nombreux types de chatbots et une façon simple de les différencier est de prendre en compte leur capacité de conversation. L'image suivante est la plus utilisée pour illustrer cette classification :



On distingue deux catégories de conversations : domaine ouvert ou domaine fermé; ainsi que deux catégories de réponses : préparées à l'avance ou générées intelligemment.

Un **chatbot** donnant des réponses prédéfinies dans un domaine libre est impossible puisqu'il faudrait pour cela avoir anticipé chaque scénario possible.

Un **chatbot** donnant des réponses prédéfinies dans un domaine limité représente la base des chatbots et est relativement aisé à mettre en place. Attention : il est néanmoins possible d'approcher fortement l'intelligence, mais tout doit être anticipé et hardcodé.

Les **chatbots** donnant des réponses générées dans un domaine restreint ont été rendu possibles par l'évolution des technologies de Machine Learning. Selon le domaine et la qualité voulue la difficulté peut varier grandement.

Les **chatbots** donnant des réponses générées dans un domaine quelconque sont très difficile à mettre en place et sont sujets à des recherches au sein des plus grandes entreprises du numérique. Le but de ces chatbots est d'arriver à suivre n'importe quelle conversation avec un humain, et aussi bien (voir mieux) qu'un être humain.

# Principe général de fonctionnement

Un des enjeux des chatbots est de parvenir à comprendre les messages de l'utilisateur de façon fiable. On se rendra compte aisément que chercher si un mot est dans le message pour déclencher une réponse prédéfinie atteint rapidement ses limites. La compréhension du langage est un large champ de recherche à lui tout seul et il existe des services dits de NLU (Natural Language Understanding) que l'on peut rendre capables de repérer l'intention (*intent*) de l'utilisateur et les paramètres (*entities*) de cette intention.

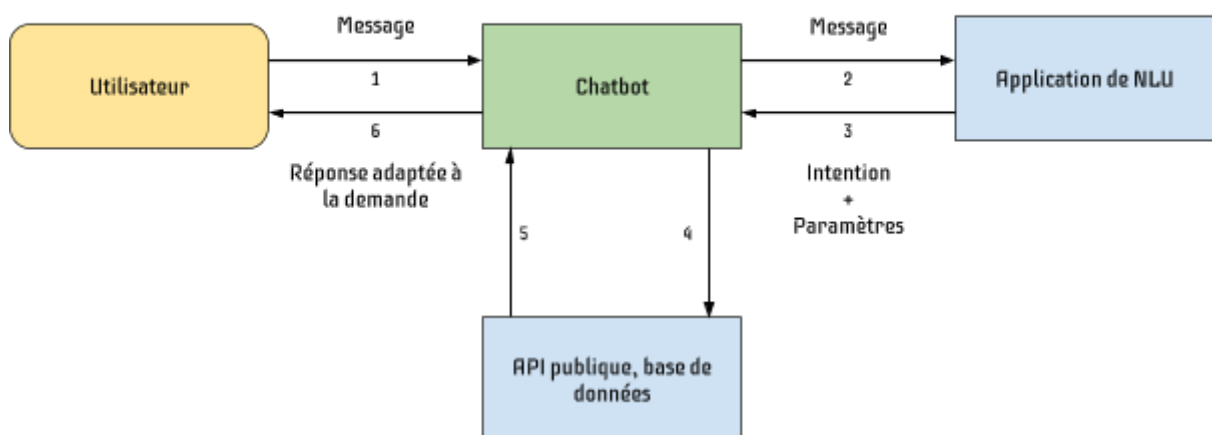
## Utilisation d'un service de NLU



En effet le but n'est pas de comprendre ce que l'utilisateur a dit, mais de comprendre ce qu'il désire. Pour être efficace, il faut entraîner l'app de NLU en lui fournissant des messages et en lui indiquant quels sont les intentions et les paramètres dans les messages.

Une fois la compréhension de l'utilisateur possible, il suffit par exemple d'aller chercher une pizzeria dans une base de données et de passer une commande.

## Fonctionnement simplifié d'un chatbot



# Blocages classiques et solutions

## Mécanisme de retry :

Il peut arriver que le chatbot ait mal compris le message de l'utilisateur, il faut alors que l'utilisateur ait la possibilité de reformuler son message. Cependant, on ne peut pas se permettre de demander à l'utilisateur si il est satisfait à chaque fois que le chatbot envoie une réponse sans rendre l'expérience utilisateur mauvaise. Personne n'a envie de doubler tous ses messages d'un *"La réponse était satisfaisante"* même si il suffit de cliquer sur un bouton.

Il faudrait donc disposer d'un moyen d'évaluer la fiabilité de la compréhension du message puis définir un seuil à partir duquel on considère que le message n'a pas été compris pour proposer à l'utilisateur de reformuler son message.

## Messages consécutifs :

Un utilisateur peut très bien envoyer plusieurs messages consécutifs avant que le chatbot n'ait répondu au premier. Je pense qu'il faut prendre parti ici : **ou (1)** l'on considère que plusieurs messages consécutifs correspondent à autant *"d'instructions"* et il faut donc porter attention à leur ordre de traitement **ou (2)** l'on considère que plusieurs messages consécutifs forment une seule et unique *"instruction"*.

Problème avec **(2)** : peut être que la réponse n'a pas été envoyée, mais qu'elle est en cours de traitement, ce qui rendrait impossible la *"fusion"*. On pourrait envisager de mettre un délai avant de commencer à traiter la requête mais cela risque de rendre l'expérience désagréable.

Problème avec **(1)** : il faut bien maîtriser tous les éléments utilisés (la plateforme, la structure du code) pour être certain que tous les messages seront traités dans l'ordre et que les réponses seront dans l'ordre également. S'ajoute à cela la possibilité que des accès simultanés en base de données peuvent générer des conflits.

## Gérer les rappels :

Souvent dans les fonctionnalités d'un chatbot figure un système de rappels mais un tel système peut être compliqué à gérer en raison des différents fuseaux horaires si ils ne sont pas gérés. Si un utilisateur demande à ce qu'un rappel lui soit envoyé à 20h, il veut le recevoir à 20h heure de là où il vit et non pas à 20h heure de chez le développeur ou 20h heure du serveur du chatbot. Il faudrait donc d'une manière ou d'une autre déterminer puis stocker le fuseau horaire de l'utilisateur.







Plutôt que de demander à l'utilisateur son fuseau horaire, il pourrait être envisageable de demander à l'utilisateur dans combien de temps il veut recevoir un rappel. Il serait alors possible de décrémenter un compteur toutes les X minutes jusqu'à atteindre 0 et envoyer le rappel mais cela demanderait dans la majorité des cas un travail supplémentaire de la part de l'utilisateur, ce qui est le contraire de ce que l'on veut accomplir avec un chatbot.

Je pense qu'il est préférable de demander à l'utilisateur à sa première utilisation.

## Utilisateur *"malveillant"* :

Il arrivera qu'un utilisateur ne réponde pas au chatbot de façon appropriée. Un chatbot devrait être capable de s'en rendre compte et d'agir en conséquence plutôt que de répondre d'un façon qui sera nécessairement erratique.

## Les plateformes de messagerie




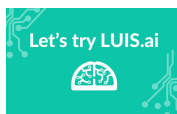

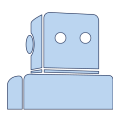
Plateforme	Utilisateurs actifs	Type de bot	Usage	Restrictions particulières
Discord 	45 M / mois (mid 2017)	Intégré à un serveur	Facilitateur, récréatif	Doit être ajouté au serveur par un utilisateur ayant les droits
Messenger 	1,2 B / mois (mid 2017)	Page Facebook	Tous	
Skype 	300 M / mois (fin 2016)	Skype app	Tous	Ne peut pas lire de fichiers
Slack 	5,8 M / semaine (fin 2016)	Intégré à un serveur	Facilitateur, récréatif	Doit être ajouté au serveur par un utilisateur ayant les droits
Telegram 	100 M / mois (deb 2016)	Bot user	Tous	
Twitter 	328 M / mois (deb 2017)	Twitter app	Tous	

Messenger semble être la plateforme la plus profitable pour héberger un chatbot, de son important nombre d'utilisateurs réguliers et de sa facilité d'accès.

Les chatbots sur Discord et Slack semblent avoir une portée moindre puisque les bots sont intégrés à des serveurs : pour communiquer avec un chatbot il faut aller sur un serveur où il est installé ou alors l'installer sur un de ses serveurs.

## Tableau récapitulatif des services de NLU

NLU	Maison mère	Coût pour 10 000 requêtes	Plateformes	Interface	Langages	Autre
-----	-------------	---------------------------	-------------	-----------	----------	-------

 Amazon Lex	Amazon	7,5 USD	Messenger et Slack (au moins)	Console	?, peut se porter sur mobile	<a href="#">Pas disponible en Europe</a>
 api.ai	Google	Gratuit	Messenger, Skype, Slack, Telegram, Twitter	Web	SDK pour : JS, Node.js; Ruby, C#, C++, Python, PHP, Java, Android, IOS	Beaucoup de doc et de samples
 IBM Watson	IBM	Gratuit jusqu'à 10000 et limité en intent et entities	Messenger et Slack (au moins)	Console	SDK pour : Node.js, Java, .NET, Python	
 Let's try LUIS.ai	Microsoft	Gratuit jusqu'à 10000, 7.5 USD / 10000 supplémentaires	Toutes	Web	Node.js, .NET, REST	Microsoft Bot Framework (voir plus bas)
 RASA	RASA	Gratuit	Toutes	Console	Python, Tous	Tourne en local ⇒ pas de HTTPS à envoyer Bonne doc
 Wit.AI	Facebook	Gratuit	Surtout Messenger	Web	SDK : Node.js, Python, Ruby	API HTTP Bonne doc

Amazon Lex et IBM Watson se distinguent par leur manque de doc.

Luis implique d'utiliser le Microsoft bot framework.

Rasa a le mérite d'être en local et n'est pas détenu par une grosse entreprise.

Api et Wit sont proches : gratuité totale, beaucoup de doc, interface web.

## Microsoft bot framework

Microsoft met à disposition des développeurs divers services pour leur faciliter la création d'un chatbot, ceci en est un bref tour d'horizon.



Un SDK -Software Development Kit- open source est à disposition en .NET ou en Node.js. Il a pour but de faciliter le développement et de généraliser le code des chatbots. Avec ce SDK sont fournis beaucoup d'exemples de bots, plus ou moins un par feature du SDK.

Microsoft propose de déployer tout bot écrit avec ce SDK vers diverses plateformes de messagerie -Courriel, SMS, Messenger, Skype, Slack et Telegram notamment- à partir de leur plateforme.

Il est possible de télécharger le Bot Framework Emulator pour tester et déboguer son bot en local.

Microsoft dispose également de nombreux services qui peuvent être utiles à un chatbot : Azure Search (Search as a Service), QnA Maker permet de transformer une FAQ en un chatbot simple, Computer Vision API permet de faire de la reconnaissance d'images.