Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

heig-vd
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

HEIG-VD

Travail de Bachelor

Guichet d'informations virtuel : conception et évaluation d'un chatbot touristique

Gabriel Luthier

Supervisé par
Prof. Andrei Popescu-Belis

Mandaté par Lausanne Tourisme, représenté par
Emmanuelle Rose

27 juillet 2018

Résumé

Mandaté par Lausanne Tourisme, ce projet propose un prototype d'agent conversationnel dont la tâche consiste à répondre aux questions posées par des touristes concernant toute activité touristique à Lausanne et environ. Avoir un guichet d'information interactif représente un avantage certain, car il n'est pas lié aux heures d'ouverture et est donc disponible non-stop.

Afin d'accomplir cette tâche, trois différentes méthodes de recherche de réponse ont été mise en place. La première consiste à analyser la question à l'aide de méthodes de traitement automatique du langage naturel pour y ressortir des entités spécifiques et proposer ensuite des réponses ciblées en fonction. La deuxième consiste à aller chercher une question similaire dans une base de données contenant des paires de question/réponse (FAQ) pour retourner la réponse associée si les questions correspondent. La dernière méthode consiste, de manière similaire, à aller chercher une entrée semblable dans une base de données contenant des points d'intérêt (provenant du site de Lausanne Tourisme) composés d'un titre et d'un descriptif puis de les retourner si la question semble correspondre à l'entrée. Ensuite l'application s'occupe d'élire la méthode la plus apte à répondre à la question posée et retourne alors la réponse associée.

Avec à cette architecture, le chatbot fourni des résultats satisfaisants. Bien qu'assez efficace, plusieurs éléments peuvent cependant être améliorés. Des piste d'améliorations sont ensuite analysée, dans l'éventualité d'utiliser un tel agent en production. Une première approche implique simplement améliorer chaque composant du système et fournirait alors de meilleures performances avant d'atteindre une limite d'évolution. Une seconde approche consiste inversement à limiter le champ d'action de l'agent pour le rendre plus performant uniquement pour des questions techniques (stockées dans la FAQ). Puis une dernière approche consiste à modifier tout le système pour mettre en place un réseau de neurones se chargeant de décider quelle méthode de réponse convient le mieux à la question posée.

Globalement fournissant des performances intéressantes, ce projet comporte plusieurs moyens d'amélioration, surtout au vu des avancées récentes de l'état de l'art dans ce domaine. Cette approche combinant plusieurs méthodes de réponse au lieu de se concentrer à en améliorer une en particulier a un potentiel de fournir d'excellents résultats, à condition de faire évoluer le projet.

Table des matières

Table des figures

Liste des tableaux

1 Introduction

En 1966 est développé *ELIZA*, l'un des tous premiers programmes informatiques qui s'apparente à un agent conversationnel (aussi appelé *chatbot*). Doté d'un fonctionnement assez simple reformulant les affirmations de l'utilisateur en questions dans le but de simuler un psychothérapeute, *ELIZA* a néanmoins ouvert les portes au développement de chatbots.

Au fil des années, les programmes se sont améliorés, motivés par la compétition du test de Turing qui consiste à mettre en confrontation une personne face à un humain ou à un ordinateur. Si elle n'arrive pas à définir avec qui elle est en train d'interagir, alors on peut en conclure que l'ordinateur a passé le test.

Au delà de la recherche du traitement automatique du langage naturel (*TALN*), une volonté d'améliorer les interfaces utilisateurs complexes pousse aussi au développement de tels systèmes. En effet, deux forces majeurs des chatbots concernant l'UX sont 1) de permettre à l'utilisateur d'écrire ce qu'il veut au lieu de naviguer à travers une interface potentiellement surchargée et 2) d'humaniser l'interaction avec le programme informatique pour obtenir une utilisation plus agréable.

La popularité des chatbots a fortement augmenté ces dernières années avec l'avènement des assistants vocaux développés par les grandes firmes qui comportent en plus des chatbots classiques la capacité de traitement de la parole pour améliorer encore plus l'interaction homme-machine.

But du projet Malgré tout, aucun système n'a actuellement convaincu unanimement le public et une forte progression dans le milieu est encore faisable. C'est dans ce contexte que se place ce projet. Mandaté par *Lausanne Tourisme*, le but est de créer un chatbot capable de répondre à des questions relatives au milieu touristique à Lausanne puis de l'évaluer pour obtenir une analyse des performances et une estimation des futures améliorations possibles en termes de coûts et de bénéfices pour les utilisateurs.

Plan du rapport La section 2 présente un aperçu de l'état de l'art des systèmes de questions-réponses. La section 3 montre quelques exemples de chatbots existants. La section 5 défini la spécification complète du projet. Puis la section suivante explique comment les données utilisées ont été formatée pour pouvoir les utiliser comme source de connaissance par le chatbot. La section 7 explicite l'architecture du système. La section 8 expose la manière dont l'agent conversationnel a été développé. Ensuite la section 9 montre différents exemples de comportement du chatbot en action. La section 10 met en valeur les différents tests qui ont été effectués pour évaluer les performances, puis la section 11 fait le point sur les résultats du système. La section 12 propose différentes améliorations du système actuelle. La section 13 expose une généralisation du fonctionnement du contrôleur du chatbot. La section 14 donne des perspectives de développement futur du projet. Pour finir, la section 15 fait le point sur le projet.

2 Aperçu de l'état de l'art

L'interaction humain-machine est un domaine qui a beaucoup évolué au cours de ces dernières années grâce à des nombreuses innovations du traitement automatique du langage naturel en, dans un premier temps, combinant plusieurs modules tels que l'extraction d'information (*IE : Information Extraction*) ou encore la recherche d'information (*IR : Information Retrieval*) (Harabagiu et al., 2000 [1]). Une nouvelle application avancée du domaine est ainsi mise en place, les systèmes de questions-réponses à domaine ouvert (*Open-Domain Question Answering Systems*), qui a pour but non seulement de retrouver quel document pourrait contenir la réponse, mais aussi de chercher le passage exact dans le texte. Une étude des performances à été faite par Moldovan et al. en 2003 [2] et montre que ces systèmes sont performants, mais limités par leur module le plus faible. Notamment la source de savoir est importante, car c'est elle qui contient toutes les données dont pourrait utiliser le système pour répondre aux questions. De bon résultats ont été obtenu en utilisant Wikipédia comme source (Ryu et al., 2014 [3]).

Dans un deuxième temps, l'intégration de techniques d'apprentissage automatique (*machine learning*) a permis une amélioration des performances globales des systèmes de dialogue non-finalisé (cherchant à imiter une conversation entre humain, sans avoir pour but d'accomplir une tâche précise). En 2015, Vinyals et Le [4] ont proposé un modèle capable de générer des conversations simples et d'extraire des données d'une source contenant du bruit à l'aide d'un réseau de neurones récurrents (RNN). Leur modèle est capable de prédire la phrase suivante en fonction de la précédente et arrive à former des réponses à plusieurs types de questions, mais n'arrive néanmoins pas à fournir une conversation réaliste. Serban et al. proposent en 2016 [5] un système de dialogue utilisant un modèle de réseau génératif qui produit des réponses mot par mot ouvrant ainsi la possibilité de dialogue réaliste et d'interaction flexible.

Une combinaison des différentes techniques semble être la solution qui donne les meilleures résultats. Une approche combinant de la recherche basée sur du hachage de bigrammes et de l'appariement TF-IDF avec un modèle de RNN entraîné à détecter des paragraphes dans Wikipédia fournit de bons résultats (Chen et al., 2017 [6]). Une autre stratégie de recherche est explorée pour combiner l'entrée de l'utilisateur avec la base de données contenant toutes les termes existants et, en parallèle, une représentation par vecteur de l'historique du dialogue actuel est comparé avec des représentations par vecteur d'historiques de dialogues complets (Banchs et Li, 2012 [7]). Cette recherche dual permet donc de prendre en compte le contexte de la discussion pour être plus pertinent dans la discussion.

Un sous-problème d'interaction humain-machine est les conversations à texte court (*short text conversations*) et consiste à répondre au message qu'un humain envoie à la machine. Une bonne source de savoir réside dans les réseaux sociaux de micro-blogging tels que Twitter [1] ou encore Weibo [2] et permet au système de se comporter intelligemment grâce aux grandes quantités de données (Ji et al., 2014 [8]). Encore une fois, l'utilisation de RNN permet d'améliorer encore plus les performances actuelles. Shang et al., 2015 [9] obtiennent plus de 75% de réponses appropriées avec de telles techniques.

1. `https://twitter.com/`
2. `https://www.weibo.com`

3 Chatbots existants

Ces dernières années ont vu l'avènement des assistants vocaux comme *Siri* (Apple), *Cortana* (Windows) ou encore *Alexa* (Amazon). Permettant une conversation réaliste tout en étant capables d'exécuter différentes tâches comme l'envoi de messages ou encore l'ajout d'un événement à un calendrier, ces systèmes sont malheureusement fermés et donc leur règles de fonctionnement restent inaccessibles.

Facebook a permis au public la création aisée de chatbots avec leur application de messagerie *Messenger*. Ayant racheté *WIT.AI*, ils bénéficient ainsi d'une solution de traitement automatique du langage naturel. D'autres moteurs de traitement sont aussi disponibles tels que *DialogFlow* ou encore *LUIS.AI*.

Beaucoup de bots sur Messenger on vu alors le jour, grâce à la facilité d'intégration au service de Facebook. En voici quelques exemples :

- **Instalocate**[3] : analyse les vols d'avions pour recevoir des informations concernant les retards et donne des directives pratiques en cas d'annulation
- **Erwin**[4] : pose des énigmes et donne des pistes si l'utilisateur ne trouve pas
- **NewsBytes**[5] : permet d'obtenir des synthèses de l'actualité
- **Langbotic**[6] : entraîne l'apprentissage du chinois à travers des exercices de traduction et de conversation

Généralement les performances des bots dits de dialogue finalisé (donc orientés à l'exécution d'une tâche particulière) sont plutôt bonnes si on n'essaie pas de les faire échouer. Les bots à but conversationnel sont pour leur part moins satisfaisants. Celui du *Wallt Street Journal* a notamment été testé sans montrer de résultats convaincants. Il n'est d'ailleurs plus disponible actuellement (sans que la raison soit connue).

4 Outils

Pour développer ce projet certain outils ont été utilisé afin d'accélérer le prototypage et donc d'obtenir un système complet comportant une variété de techniques pour l'accomplissement de sa tâche qui consiste à répondre aux questions posées par l'utilisateur, portant sur le domaine du tourisme à Lausanne. Cette section s'occupe de les présenter brièvement.

4.1 Facebook Messenger

Facebook Messenger[7] permet de concevoir un bot qui sera accessible via Facebook. Élément essentiel à ce chatbot, il sous-tend en effet tout le système. Chaque composant utile au traitement de la question puis à la formulation de la réponse est lié à cette plateforme qui, de son côté, s'occupe de gérer les interactions avec l'utilisateur.

3. `https://www.messenger.com/t/instalocate`
4. `https://www.messenger.com/t/erwin.chat`
5. `https://www.messenger.com/t/NewsBytesApp`
6. `https://www.messenger.com/t/langbotic`
7. `https://www.messenger.com/`

C'est donc avec cette plateforme que l'on peut notamment définir des messages d'accueil, des propositions de messages pour inciter l'interaction, des réponses structurées avec des images et des liens sur des sites internet. Agréable à utiliser, elle est néanmoins à double tranchant comme le montre les avantages/désavantages présentés ci-dessous.

Avantages :

- Interface utilisateur déjà fournie
- Gestion des droits d'accès efficace pour la consultation et modification du chatbot
- Possibilité d'ajouter différents produits (mettre en place un traitement automatique du langage naturel (TALN) avec WIT.AI par exemple)
- Simple à mettre en place
- Documentation bien fournie
- Automatiquement accessible sur ordinateur et smartphone

Désavantages :

- Système couplé à Facebook (il faut avoir un compte chez eux pour pouvoir interagir avec le chatbot)
- Pas possible de modifier l'affichage des réponses au-delà des réponses formatées fournies
- Dépendant des conditions d'utilisation de Facebook (notamment concernant la protection des données)

Avec la fuite de données provenant de l'affaire *Facebook-Cabridge Analytica*, l'entreprise a décider d'inspecter de manière plus prononcée la façon dont les applications connectées à Facebook utilisaient les données des utilisateurs. Au milieu du développement de ce travail, la plateforme Messenger a été inutilisable pendant quelques semaines et a donc ralenti cette partie du projet.

Une autre contrainte a été pour la validation du chatbot. En développement, tester l'application ne pose pas de problème avec un compte administrateur. Par contre, pour la mettre en ligne au public (dans l'optique de faciliter la phase de test), une procédure de validation est nécessaire et demande notamment des preuves d'existence de l'entreprise gérant le chatbot, ce qui n'a pas été possible d'obtenir dans le cadre de ce projet.

4.2 WIT.AI

WIT.AI[8] est un outil qui aide à faire comprendre à la machine ce que l'utilisateur lui demande. Capable dès le départ de détecter des entités spécifiques comme des lieux, dates, emails, etc., il peut être amélioré en lui faisant apprendre d'autres entités plus adaptées aux problèmes que l'on veut résoudre. Intégré automatiquement lors de la mise en place d'un chatbot sur Facebook Messenger, son utilisation est aisée.

8. `https://wit.ai/`

Pour ajouter des entités supplémentaires, il faut entrer des expressions à la main pour mettre en place ainsi une base de données contenant les différentes manières d'utiliser l'entité que l'on veut ajouter. L'application est capable d'apprendre et de s'affiner au fur et à mesure, en validant à la main à quelle entité correspond la nouvelle interaction d'un utilisateur.

Plusieurs stratégies de recherche d'entité personnalisée sont possibles :

1. **Trait** : à utiliser quand il n'y pas d'association directe entre les mots de la phrase et la valeur de l'entité, mais qu'on a plutôt besoin de l'entièreté de la phrase pour pouvoir déterminer sa valeur (par exemple pour des intentions, des sentiments ou encore des formules de politesse)
2. **Text libre** : à utiliser quand on a besoin d'extraire une partie du message et que cette partie de texte n'appartient pas à une liste de valeurs possibles (par exemple pour un nom de contact ou le contenu d'un message)
3. **Mots-clés** : à utiliser quand la valeur de l'entité appartient à une liste prédéfinie de mots-clés (par exemple le nom d'un pays ou le numéro d'une chambre d'hôtel)

Typiquement pour toutes les formules de politesses (bonjour, au revoir et merci) mises en place pour le chatbot, la stratégie "Trait" a été utilisée, car on a besoin de la courte phrase en entier pour comprendre ce que l'utilisateur veut dire.

Une possibilité d'utilisation supplémentaire aurait été d'utiliser la stratégie "Mots-clés" pour définir tous les noms propres d'une catégorie (par exemple tous les noms de restaurant de la ville ou tous les noms des parcs) pour ensuite pouvoir réagir de manière différente en fonction de la catégorie identifiée. Mais la valeur ajoutée de cette méthode n'a pas surpassé le temps nécessaire pour la mettre en place et n'a donc pas été implémentée.

De manière générale, WIT.AI est capable de détecter efficacement les entités personnalisées, mais l'apprentissage est fastidieux, car il doit être fait à la main. L'utilisation plus immédiate de cet outil est de l'utiliser pour détecter les entités prédéfinies qui sont plutôt de types numéraires, comme l'âge d'une personne, une quantité d'argent, une distance, une durée ou encore une température.

4.3 LunR

LunR[9] est une librairie effectuant la recherche d'information dans un corpus. Il est possible de le configurer pour qu'il ignore les mots vides (*stop word*), qu'il applique la racinisation aux mots restants à l'aide de l'algorithme Porter Stemmer[10], qui est relativement efficace en anglais, et de lui donner des poids dans la recherche des termes.

Chaque recherche retourne un score de pertinence qui est calculé à l'aide de l'algorithme *BM25*[11] qui fonctionne comme une fonction *TF-IDF* : plus le terme recherché apparaît dans un document unique, plus ce terme va augmenter le score du document. Par contre, plus le terme recherché apparaît dans la collection de documents, moins il va augmenter le score du document.

9. `https://lunrjs.com/`
10. `https://tartarus.org/martin/PorterStemmer/`
11. `nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html`

Ci-dessous est présenté la mise en place de l'index utile à la recherche dans la FAQ à l'aide de LunR :

```
const lunr = require("lunr");

class Faq {
  constructor(methods) {
    this.methods = methods;

    this.faqList = {};
    this.idxFaq = {};
    this.indexFaq = 0;

    let faq = this;

    fs.readFile('faqDb.json', 'utf8', function (err, data) {
      if (err) {
        return console.log("Error while reading faqDb.json file", err);
      }

      faq.faqList = JSON.parse(data);

      faq.idxFaq = lunr(function () {
        this.ref('id');
        this.field('question');
        this.field('answer');

        faq.faqList.forEach(function (q) {
          this.add({id: faq.indexFaq, question: q.question, answer: q.
              answer});
          faq.indexFaq++;
        }, this);
      });
    });
  }

  // [...]
}
```

Listing 1 – Mise en place de l'index pour effectuer une recherche dans la FAQ

4.4 Puppeteer

Puppeteer[12] est une librairie `Node.js` qui fourni une API de haut niveau pour contrôler le navigateur "sans tête" (*headless browser*) Chrome (ou Chromium) et permet de naviguer à travers des pages internet tout en exécutant du code `JavaScript` et donc permet d'accéder au contenu chargé de manière asynchrone. Très utile pour faire des tests, il a été utilisé ici pour aller récupérer des informations sur le site de Lausanne Tourisme[13], afin de simuler un accès plus conventionnel à travers une API.

Un exemple d'utilisation est donné ci-dessous :

```
const puppeteer = require('puppeteer');

(async () => {
```

12. `https://pptr.dev/`
13. `https://www.lausanne-tourisme.ch`

```
4   const browser = await puppeteer.launch();
5   const page = await browser.newPage();
6
7   try {
8     await page.goto("https://www.lausanne-tourisme.ch/en/P10661");
9
10    const title = await page.evaluate(() => {
11      const t = document.querySelector('h1');
12      if (t) return t.innerText;
13      else return '';
14    });
15
16    console.log(title);
17  catch (error) {
18    console.log(error);
19  } finally {
20    page.close();
21  }
22
23  await browser.close();
24 })();
```

Listing 2 – Exemple de récupération de titre d'une page web de site de Lausanne Toursime

4.5 Glitch

Glitch[14] permet de déployer l'application de manière aisée et gratuitement, ce qui est bien utile pour le développement/test du chatbot. Le projet, en `Node.js`, est stocké de manière sécurisée (surtout au niveau des clés d'API privées) et permet l'export du code source sur Github directement. Une console côté serveur est accessible pour permettre de debugger facilement.

Quelques limitations sont néanmoins présente, comme l'impossibilité d'écrire dans des fichiers de manière dynamique (par le code) ou encore la limite de 128 MB de l'application qui a posée problème lorsque Puppeteer a tenté d'être utiliser par l'application directement.

5 Spécification

La configuration de l'agent consiste à essayer, en parallèle, de trouver une réponse à une question posée par l'utilisateur, portant sur le domaine du tourisme à Lausanne, à l'aide de différentes méthodes. Puis de choisir laquelle est la plus pertinente pour extraire la réponse et la proposer à l'utilisateur. Il y a trois méthodes à disposition :

1. Détection de l'intention de la question et des valeurs associées
2. Recherche de la question la plus similaire dans une Foire Aux Questions (FAQ)
3. Recherche sur le site web de Lausanne Tourisme du Point Of Interest (POI) le plus pertinent.

Langue L'anglais à été choisi pour ce projet. *WIT.AI* fonctionne aussi avec d'autres langues (notamment le français), mais c'est pour l'anglais qu'il est le plus performant.

14. `https://glitch.com/`

5.1 Analyse de la requête

À l'aide de *WIT.AI*, le système analyse la question posée et tente de comprendre l'intention de l'utilisateur, par exemple savoir s'il cherche un lieu en particulier, un trajet ou encore l'horaire d'un magasin. Après cette étape, il récupère l'argument nommé recherché (le Musée Olympique ou le quartier Ouchy par exemple). Si l'intention est de trouver un lieu (la cathédrale par exemple), alors il va extraire le nom propre pour le chercher sur *Google Maps*. *WIT.AI* permet de donner un indice de confiance qui sera utilisé par la suite pour attribuer un score à ce processus.

L'analyse de la requête se résume ainsi :

- Analyse de la requête avec *WIT.AI*
- Recherche de l'intention de la question (le `TYPE`)
- Récupération des différentes valeurs (`args`) recherchés
- Action du contrôleur en fonction du `TYPE` et des `args`.

5.2 Recherche dans la FAQ

Ce processus va, à l'aide de *LunR*, chercher quelle question ou réponse dans une liste pré-définie de questions-réponses fréquentes (FAQ) est la plus similaire à la question posée par l'utilisateur en regardant la quantité de mots-clés identiques, dont certain pouvant être pondérés, permettant ainsi de donner plus d'importance à certains mots. Cette librairie retourne aussi un score calculé par l'algorithme et sera utilisé pour déterminer sa qualité.

Un travail de pré-traitement est nécessaire pour pouvoir mettre en place la FAQ :

- Rédaction/traduction des questions et réponses à partir de la liste reçue par Lausanne Tourisme
- Indexation des questions/réponses avec *LunR*

La recherche dans la FAQ se résume ainsi :

- Récupération des mots-clés intéressants définissant au mieux la question (élimination des mots vides)
- Recherche parmi les questions et réponses de la FAQ

5.3 Recherche parmi les POIs

Le site web de Lausanne Tourisme est constitué de *Points of Interest* (*POI*) qui représentent aussi bien un événement qu'un restaurant ou magasin. Malheureusement pas accessible directement par code via une API, une première idée pour pouvoir néanmoins obtenir un accès a été de transmettre la question au site de Lausanne Tourisme et ensuite de récupérer les informations que le site aurait trouvés dans sa base de données en allant les chercher dans le code source de la page reçue.

La recherche des POIs se serait alors faite ainsi :

- Délégation de la question au site de Lausanne Tourisme
- Récupération des résultats de recherche trouvés par le site

Malheureusement, comme les informations sur la page du site sont chargées de manière asynchrones, les récupérer avec un simple crawler n'est pas possible. Une solution possible aurait été d'utiliser un navigateur sans tête (Puppeteer par exemple) pour le faire dynamiquement à chaque requête, mais des limitations de taille de l'application hébergée sur Glitch ne le permettent pas.

Finalement, Puppeteer a été utilisé pour récupérer ces POIS par la force brute en essayant tous les identifiants possible des POI et si la page existait, alors les détails étaient enregistrés. Cette solution permet de donner une idée de comment un système plus complet pourrait fonctionner avec cette source de savoir, mais a le désavantage, dans l'implémentation actuelle, de ne pas suivre l'évolution de site web de Lausanne Tourisme et donc l'ajout de nouvelles entrées.

5.4 Gestion du dialogue

Bien qu'une conversation hors-sujet ne soit pas l'objectif de ce chatbot, quelques formules de politesse simples sont mises en place pour rendre l'échange plus agréable. Grâce à WIT.AI, le système peut détecter les formules basique telles que *hello*, *goodbye* ou encore *thanks* pour répondre en conséquence. Au delà de ces messages, aucun aspect conversationnel supplémentaire n'a été mis en place, car le but principal est de se concentrer sur l'accomplissement de la tâche voulue, à savoir répondre à une question posée par un touriste.

Les éléments liés à la gestion du dialogue sont les suivants :

- Message d'accueil
- Formules de politesse ("bonjour", "au revoir", "merci")
- Délégation de la requête à une vraie personne (si souhaité) par email
- Fin du dialogue

6 Formatage des données

Deux sources de savoir ont été utilisé pour mettre en place ce chatbot :

1. Les questions/réponses de la FAQ
2. Les POIs composés d'un titre et d'un texte descriptif

Les deux sources sont compilée chacune dans une simple base de donnée stockée dans un fichier `JSON`, car ainsi facilement utilisable avec du code `JavaScript`.

6.1 Les données de la FAQ

La liste de questions/réponses dont est composée la FAQ provient de Lausanne Tourisme. Une série de questions fréquemment posée aux guichets de l'office du tourisme a été recensée et compilée avec pour la plus part les réponses associés.

Comme il a été décider de développer le chatbot pour qu'il interagisse en anglais, il a fallu traduire ce document pour pouvoir l'utiliser. Ont été gardé uniquement les entrées comportant une réponse associée. Au total, la FAQ est peuplée de 56 paires de question/réponse. Comme certaines réponses contenaient un lien hypertexte, cette donnée a été ajoutée à part pour pouvoir être utilisable par l'application.

Voici un aperçu du document `JSON` final :

```
[
  {
    "question": "Is there a hop-on hop-off bus or a bus tour of the town ?
       ",
    "answer": "There are no bus tours, but you can use the regular buses
       or choose a guided walk or a Segway City Tour (https://www.lausanne
       -tourisme.ch/en/P20069/guided-tour-on-a-segway)",
    "link": "https://www.lausanne-tourisme.ch/en/Z5591/guided-tours"
  },
  {
    "question": "Where is the market ?",
    "answer": "There are open air markets on Saturdays in the centre (from
        place de la Riponne to place de la Palud and rue du Bourg), as
       well as smaller markets on Wednesday (Riponne) and Tuesdays /
       Thursdays (place Chauderon), and on Sundays at Ouchy.",
    "link": ""
  },
  [...]
]
```

6.2 Les données des POIs

Dans l'idéal, les POIs auraient été accédées dynamiquement en effectuant une recherche sur les serveurs du site web de Lausanne Tourisme lors de chaque question posée par un utilisateur. Ainsi la solution proposée aurait pu évoluer en fonction des changements effectués sur le site web de l'entreprise.

Il n'a malheureusement pas été possible d'avoir un accès à ces données à l'aide d'une API. La contrainte de taille de l'application sur la plateforme d'hébergement Glitch n'a pas permis d'utiliser Puppeteer pour aller récupérer dynamiquement ces mêmes données. Il a fallu donc mettre en place un script pour tout récupérer et en faire une base de données statiques.

Le crawler a été développé en `Node.js` pour rester dans le même environnement que le chatbot et faciliter ainsi son implémentation. Le framework Puppeteer a été utilisé pour pouvoir accéder au contenu chargé dynamiquement par les pages web. En effet, récupérer le code source n'est pas suffisant, car la plus part du contenu des pages est inclus de manière asynchrone et requière donc l'exécution de code `JavaScript` pour pouvoir y accéder.

Comme il n'existe pas de répertoire contenant un lien sur tous les POIs disponibles sur le site internet, il a fallu trouver une autre manière de tous les récupérer, ou du moins un maximum. Les URLs de chaque POI diffèrent uniquement par un identifiant numérique. En employant cette information et en utilisant une méthode de force brute, le crawler a essayé chaque identifiant allant de 1000 à 40'000 et si la page contenait de l'information, alors il la récupérait.

Au total, après plus d'une nuit d'exécution, 1334 POIs ont pu être récupérées, sans être un résultat exhaustif, car le script a été conçu pour ignorer les erreurs (notamment les coupures de connexion) et passer au suivant après un court délais. En plus du titre, lien et descriptif du POI, la latitude et longitude ont aussi été extraites pour des possibilités future de mise en place de recommandations basées par rapport à l'emplacement de l'utilisateur.

Voici un aperçu du document `JSON` final :

```
[
  {
    "name": "Festival de la Cité, Lausanne",
    "link": "https://www.lausanne-tourisme.ch/en/P1079",
    "text": "A free event in Lausanne held every year in July, the
       Festival de la Cité is a multi-disciplinary, ground-breaking
       festival, with unique creations, kids activities and a whole series
        of exciting concerts taking place over a week in different venues
       in the city. Unexpected, unconventional, lively: there are many
       words that can be used to describe the favourite summer event for
       the residents of Lausanne. [...]",
    "latitude": 46.522859569802,
    "longitude": 6.6348323700549
  },
  {
    "name": "BDFIL festival",
    "link": "https://www.lausanne-tourisme.ch/en/P1115",
    "text": "The BDFIL festival featuring Swiss and international comics
       takes place every September in Lausanne. The programme for this
       autumn cultural event includes exhibitions, competitions,
       screenings and activities for kids and older participants. In the
       heart of Lausanne, BDFIL is one of the biggest European events
       dedicated to the 9th art. Held in the Place de la Riponne, the
       event includes a vast programme of exhibitions, performances and
       installations and welcomes around 100 authors from Switzerland and
       elsewhere.",
    "latitude": 46.523351234887,
    "longitude": 6.6328687872738
```

```
15    },
16    [...]
17  ]
```

7 Architecture

Dans *Open-domain question answering from large text collections* [10], au chapitre 2, Paşca explicite une architecture générique pour la récupération de documents (avant de partir sur une architecture complète pour un système de questions-réponses à domaine ouvert de collection de large textes). Basé là-dessus, la figure 1, page 17 montre l'architecture qui a été mise en place pour le chatbot.

En premier lieu, il y a un utilisateur qui a besoin d'information. Il formule donc un question qu'il soumet au chatbot. Une analyse par le traitement automatique du langage naturelle est faite sur les serveurs de Facebook. Puis l'application va faire correspondre les meilleures documents dans la FAQ et dans les POIs. Avec ces informations provenant des trois méthodes de réponse, le contrôleur s'occupe ensuite de proposer la meilleure réponse en retour à l'utilisateur.

L'avantage de ce type d'architecture est que l'ajout de nouvelle méthode de réponse se fait facilement. En effet, à partir de la question de l'utilisateur, il suffit de la transmettre au nouveau module comme entrée et de connecter la sortie de celui-ci au contrôleur qui s'occupera à son tour de gérer toutes les réponses reçues (ainsi que leur score) pour fournir la meilleure de retour à l'utilisateur.

8 Implémentation

Cette section décrit comment les différents modules du système ont été implémentés.

8.1 Fonctions d'attribution de score

Chaque méthode permettant de fournir une réponse éventuelle comporte différents paramètres fournissant un score numérique qui est ensuite classifié dans une des trois différentes catégories indiquant à quel degré de satisfaction elle prétend avoir obtenu une réponse pertinente à la question posée :

- Élevé : la méthode a obtenu un résultat très satisfaisant
- Moyen : la méthode a obtenu un résultat qui pourrait répondre à la question, mais sans être sûr
- Faible : la méthode n'a pas réussi à obtenir une réponse assez satisfaisante

Pour chaque méthode, les fonctions d'attribution de score ont été ajusté pour satisfaire, dans chaque catégorie, un ensemble d'exemples. À partir de ses mots-clés/questions, et des scores brutes retournés par les différents outils (WIT.AI, LunR) utilisés, les seuils définissant la distribution dans les catégories ont été fixés.

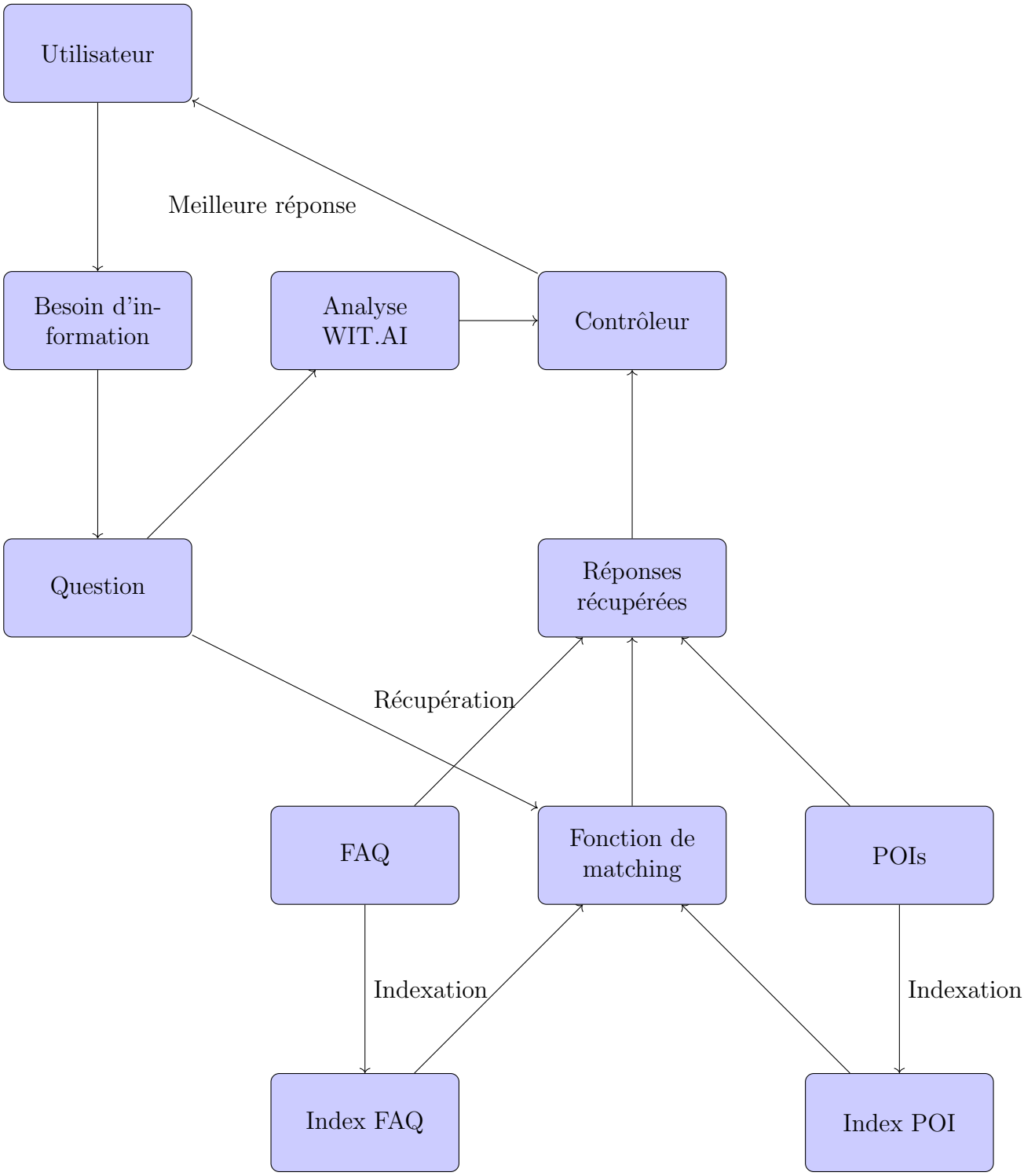

FIGURE 1 – Architecture du chatbot

À chaque méthode est attribué une priorité pour permettre au contrôleur de les départager en cas d'égalité (plus de détails à la section 8.2). Pour obtenir un ordre absolu, chaque méthode s'est vue attribuée un valeur de priorité différente.

8.1.1 Traitement automatique du langage naturel

Pour cette méthode, les exemples ont été défini ainsi :

- Élevé :
  - nom propre de lieu (*Ouchy, Flon*)
  - nom descriptif de lieu et nom propre de lieu (*Musée Olympique, Place Palud*)
  - question complète avec nom propre de lieu (*Where is located Ouchy ?*)
  - question complète avec nom descriptif de lieu et nom propre de lieu (*Where is the Art Brut Museum ?*)
- Moyen :
  - nom de lieu vague (*beaches, restaurants*)
  - question annexe contenant un nom propre de lieu (*What is the schedule of the boat to go to Evian ?*)
  - question annexe contenant un nom descriptif de lieu (*Where are the toilets near the Cathedral ?*)
- Faible :
  - nom propre autre que de lieu (*Federer, Jupiter*)
  - question hors-sujet (*How is the weather like today ?*)
  - mots-clés de style commande (*events, bus schedule*)

WIT.AI retourne un indice de confiance entre 0 et 1 qui indique à quel point il pense que l'entrée contient une entité donnée (plus l'indice est proche de 1, plus l'analyse est confiante que l'entrée contient l'entité). Cet indice a donc été utilisé pour définir les deux seuils permettant d'attribuer la catégorie indiquant le degré de satisfaction de la réponse.

8.1.2 FAQ

Pour la FAQ, les exemples ont été définis simplement en analysant le degré de similarité entre la question posée par l'utilisateur et l'entrée dans la FAQ. C'est ce que fait LunR lors de la recherche, mais ces exemples ont été défini pour pouvoir ensuite définir quel seuil du score de LunR donnait quelle catégorie.

- Élevé :
  - question identique à une entrée de la FAQ (*How can I go by foot from the station to the lake ?*)
  - question avec les mots-clés principaux identiques (*go foot station lake*)
  - plusieurs mots-clés contenus dans une réponse d'une entrée de la FAQ (*reach lake shore from station*)

- Moyen :
  - peu de mots-clés principaux identiques (*station lake*)
  - peu de mots-clés principaux identiques dans une réponse (*reach lake*)
  - question reformulée avec les mêmes mots-clés principaux (*How long does it take to go on foot to the station from the lake ?*)
- Faible :
  - question n'apparaissant pas dans la FAQ (*How is the weather like today ?*)
  - mots-clés n'apparaissant pas ensemble dans des entrées de la FAQ (*lake restaurant recommendation*)
  - mots-clés génériques d'une question/réponse (*how can go station*)

Pour la FAQ, LunR nous donne un score (qui n'est pas compris dans un intervalle fixe) donnant la pertinence du résultat retourné. Plus le score est élevé, plus le résultat est pertinent. Pour attribuer le score total, une combinaison de plusieurs critères a été fait :

1. **Score** :
   - Élevé : 4 et plus
   - Moyen : entre 2 et 4
   - Faible : entre 0 et 2
2. **Différence entre le 1er et le 2ème résultat** :
   - Élevé : 4 et plus
   - Moyen : entre 1.5 et 4
   - Faible : entre 0 et 1.5
3. **Démarcation du 1er et 2ème par rapport au 3ème résultat** :
   - Élevé : les 1er et 2ème sont à plus de 4 de différence du 3ème
   - Moyen : les 1er et 2ème sont entre 1.5 et 4 de différence du 3ème
   - Faible : les 1er et 2ème sont à moins de 1.5 de différence du 3ème
4. **Longueur de la question posée** :
   - Élevé : 3 mots ou plus
   - Moyen : entre 2 et 3 mots
   - Faible : entre 0 et 1 mot

Puis le total de tous les sous-score est additionné pour finalement être réparti ainsi :

- Élevé : 3 et plus
- Moyen : entre 2 et 3
- Faible : entre 0 et 2

La logique derrière cet algorithme peu intuitif est la suivante :

1. **Score** : de manière immédiate, plus le score du premier résultat est élevé, plus il est pertinent
2. **Différence entre le 1er et le 2ème résultat** : si le premier résultat se démarque du deuxième, cela renforce la légitimité
3. **Démarcation du 1er et 2ème par rapport au 3ème résultat** : dans la même logique, mais avec le groupe du premier et deuxième ensemble de démarquant du troisième résultats retourné
4. **Longueur de la question posée** : plus la question est longue (et donc ressemble plus à une vraie question au lieu d'une sorte de commande), plus la premier résultat retourné a des chances d'être pertinent, car il peut se baser sur plusieurs mots-clés lors de la recherche

Ces différents paramètres ont été mis en place pour affiner au mieux la qualité du résultat retourné. Il aurait aussi été possible, afin d'obtenir une logique plus simple et moins prône aux erreurs, d'uniquement prendre en compte l'indice de score fourni par LunR pour le premier résultat. On aurait alors perdu le contexte entourant cette première réponse retournée, c'est pour cela que la version complexe a été mise en place.

8.1.3 POIs

Pour les POIs, les exemples ont été défini ainsi :

- Élevé :
  - mots-clés identique au titre d'un POI (*Lausanne Cathedral*)
  - beaucoup de mots-clés apparaissant dans la description du POI (*old town cathedral gothic*)
  - question contenant beaucoup de mots-clés apparaissant dans la description du POI (*What is the spiritual capital of French-speaking Switzerland ?*)
- Moyen :
  - mots-clés partiel du titre d'un POI (*Cathedral*)
  - quelques mots-clés apparaissant dans la description du POI (*old town*)
  - question contenant quelques mots-clés apparaissant dans la description du POI (*Where is the spiritual capital ?*)
- Faible :
  - mots-clés pas contenu ni dans le titre ni dans la description du POI
  - question dont presque aucun mots-clés n'apparait ni dans le titre ni dans la description du POI

Pour attribuer un score à la recherche des POIs, uniquement le score retourné par LunR a été utilisé, car il était prévu à la base d'obtenir des paramètres permettant d'affiner le score lors de la recherche sur le site de Lausanne Tourisme. Une logique similaire à la recherche dans la FAQ pourrait aussi être mise en place.

8.2 Algorithme du contrôleur

L'algorithme du contrôleur est assez simple et est facilement explicable par écrit :

1. Il effectue la recherche de réponse avec toutes les méthodes à disposition
2. Les trie par score
3. En cas d'égalité, départage les méthodes par priorité la plus élevée
4. Retourne la meilleur méthode trouvée
5. Si l'utilisateur n'est pas satisfait, retourne au fur et à mesure les autres réponses trouvées par les méthodes (et donc toujours classifiées par score, puis par priorité)

L'avantage de cet algorithme est qu'il est facile d'ajouter d'autres méthodes de réponse à l'application. En effet, il suffit de lui donner une priorité par rapport aux autres méthodes déjà utilisées et le contrôleur pourra continuer à fonctionner sans aucune modification. Un autre avantage est qu'il est facilement compréhensible et diminue donc les erreurs d'implémentation, à l'instar des fonction d'attribution de score des différentes méthodes de réponse.

9 Exemples de dialogues

Dans les captures d'écran présentées dans cette section, les bulles de texte sur fond bleu situé à droite de l'image correspondent au messages écrit par l'utilisateur (ou à des messages envoyés automatiquement lors de l'appui d'un bouton par l'utilisateur) et les bulles de texte sur fond gris situé à gauche de l'image correspondent aux réponses du chatbot.

9.1 Message d'accueil

La figure 2, page 22 montre le message de salutation lorsque l'utilisateur interagit pour la première fois avec le chatbot. Une message de bienvenue est affiché avec un bouton "Démarrer" dont il est nécessaire d'activer pour pouvoir utiliser le bot.

Une fois le bouton appuyé, l'utilisateur est accueilli par un message lui proposant de poser n'importe quelle question (figure 3, page 23).

9.2 Chit-chat

Même s'il ne s'agit pas du but premier du chatbot, un minimum de réaction aux formules de politesse basique a été mis en place. Il est donc capable de réagir aux salutations, remerciements et aux au revoir, comme le montre la figure 4, page 24.

9.3 Interaction avec le chatbot

Un exemple de conversation où le système arrive à trouver une réponse au bout de la 3$^{\text{ème}}$ tentative est montré à la figure 5, page 25. On voit ainsi l'interaction principale de la part de l'utilisateur quand il n'est pas satisfait d'une réponse. Il clique alors sur le bouton `Not what I want` qui indique alors au chatbot qu'il doit essayer une autre méthode (si possible).

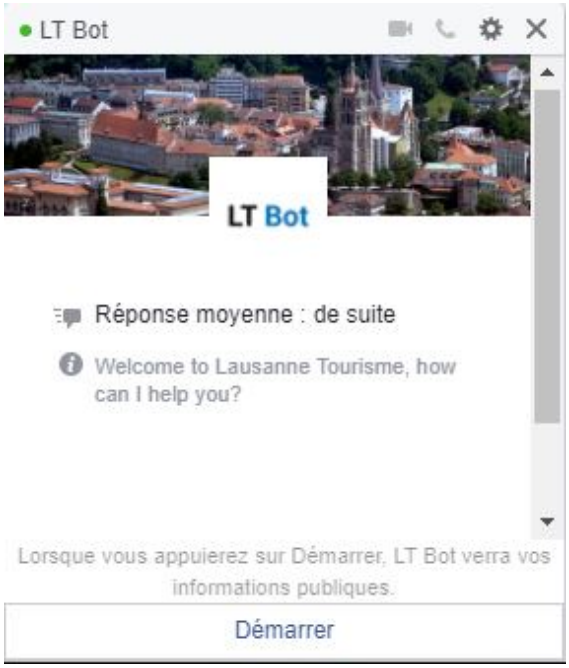

FIGURE 2 – Message de salutation lors de la première interaction avec le bot

Si le chatbot n'arrive pas à fournir une réponse, il va proposer à l'utilisateur de contacter le staff pour que quelqu'un puisse lui répondre manuellement (figure 6, page 26). En choisissant cette option, un envoi d'email contenant la question posant problème est effectué pour informer un employé de la situation.

Certaines entrées de la FAQ contiennent un lien hypertexte, la figure 7, page 26 montre comment le système les gère. Il ajoute simplement un bouton `Further informations` qui permet d'ouvrir une page web contenant les informations supplémentaires.

Concernant les demandes de recommandation, le chatbot n'est pas toujours assez pertinent. Il lui arrive de simplement retourner un élément lorsqu'on lui demande une liste de recommandation (figure 8, page 27). Dans d'autres cas (généralement quand il y a une entrée dans la FAQ), il arrive à retourner quelques propositions (figure 9, page 27).

10 Tests

10.1 Recherche de la méthodologie adéquate

Dans l'apprentissage automatique (*machine learning*), l'évaluation d'un programme se fait en partageant en deux les données d'apprentissage. Le premier ensemble, appelé jeu de données d'apprentissage, permet à l'algorithme de modéliser le problème en ajustant ses paramètres en fonction des données lues. Le deuxième ensemble, appelé jeu de données de test, s'occupe de tester si le modèle construit par l'algorithme fonctionne bien sur des données nouvelles, pas encore utilisées pour l'apprentissage.

Cette méthode de test est efficace, mais souffre d'un désavantage majeur : elle requiert une base de données conséquente pour permettre un apprentissage pertinent suivi d'un test couvrant un maximum de cas. Dans le cadre de ce projet, une centaine de questions provenant

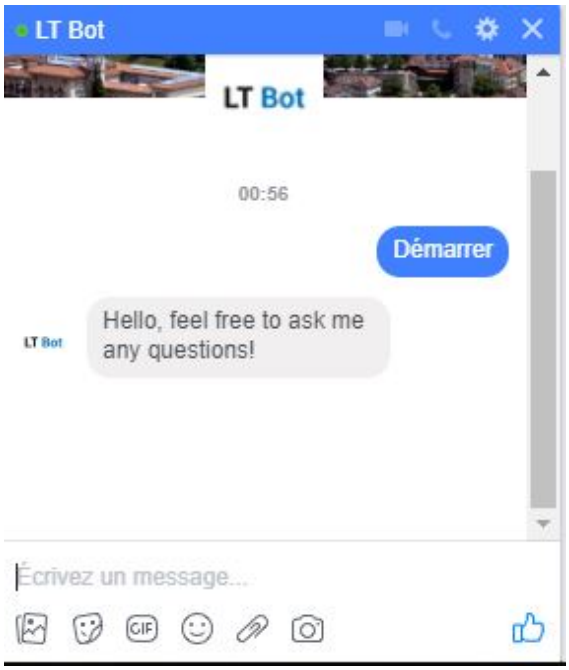

FIGURE 3 – Message encourageant l'utilisateur à poser une question au bot

de cas concrets recueillis par Lausanne Tourisme sont à disposition, mais ce n'est pas suffisant pour entrainer et tester de façon correcte un système.

Une autre méthode d'évaluation consiste à demander à des utilisateurs cibles (dans notre cas de vrais touristes visitant Lausanne) de tester le programme. La difficulté de cette méthode consiste à trouver assez de personnes ayant des besoins variés pour permettre de tester au maximum les capacités du chatbot, et demande trop de temps et de ressources pour ce projet.

Une variante de cette méthodologie a été initialement prévue et consistait à demander à des employés de Lausanne Tourisme dont le travail consiste (notamment) à répondre aux questions des touristes, de venir tester les performances du chatbot. L'avantage de cette procédure est que ces testeurs ont une connaissance globale des questions types posées par des touristes et pourraient donc se projeter facilement dans la peau de l'un d'eux. Malgré tout, il ne s'agit pas d'une méthode parfaite, car ayant connaissance des réponses, ils risquent de ne pas poser les questions de la même manière que le ferait un touriste.

Pour des raisons de logistique, il n'a pas été possible d'effectuer les tests ainsi. Au lieu des employées de Lausanne Tourisme, 15 personnes sont venues tester le chatbot et chacune a posé trois questions, en s'imaginant à la place d'un touriste venant visiter Lausanne. Ce groupe de testeur ne représente pas très bien un touriste lambda, car il est composé uniquement de jeunes entre 22-28 ans connaissant déjà la ville. Malgré tout, ils ont posé des questions qui n'avaient alors pas encore été testées et ont donc pu apporter un regard nouveau sur les capacités du chatbot.

10.2 Procédure des tests

Comme indiqué ci-dessus, chacun des 15 participants a posé trois questions au chatbot puis a jugé l'interaction. La procédure trop compliquée de validation de l'application par

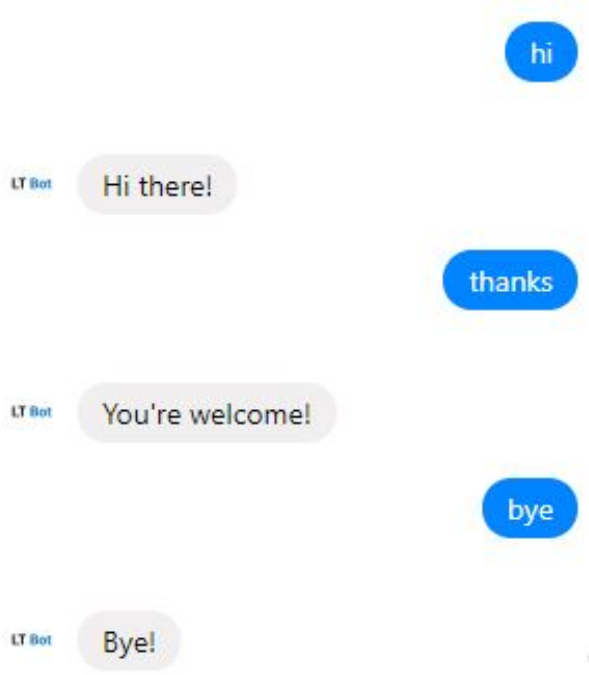

FIGURE 4 – Messages de conversation polie auxquels le bot est capable de comprendre et répondre

Facebook a fait qu'il a fallu tester l'agent sur le compte administrateur du bot (le miens), donc en ma présence. Cela a impliqué que les participants n'ont pas vraiment fait de remarques ou commentaires et donc le champ prévu à cet effet a alors été rempli par la méthode de réponse qui a fourni ce que l'utilisateur cherchait. La totalité des résultats des tests est disponible à l'annexe D.

Pour chaque question posée, le participant a répondu aux points suivants :

- Quelle a été la question posée ?
- Avez-vous été satisfait de la réponse ?
  - Très satisfait
  - Plutôt satisfait
  - Plutôt insatisfait
  - Très insatisfait
- Après combien de tentatives du chatbot avez-vous obtenu une réponse ?
  - 1
  - 2
  - 3
  - plus
  - jamais
- Avez-vous des remarques, commentaires, ... ?

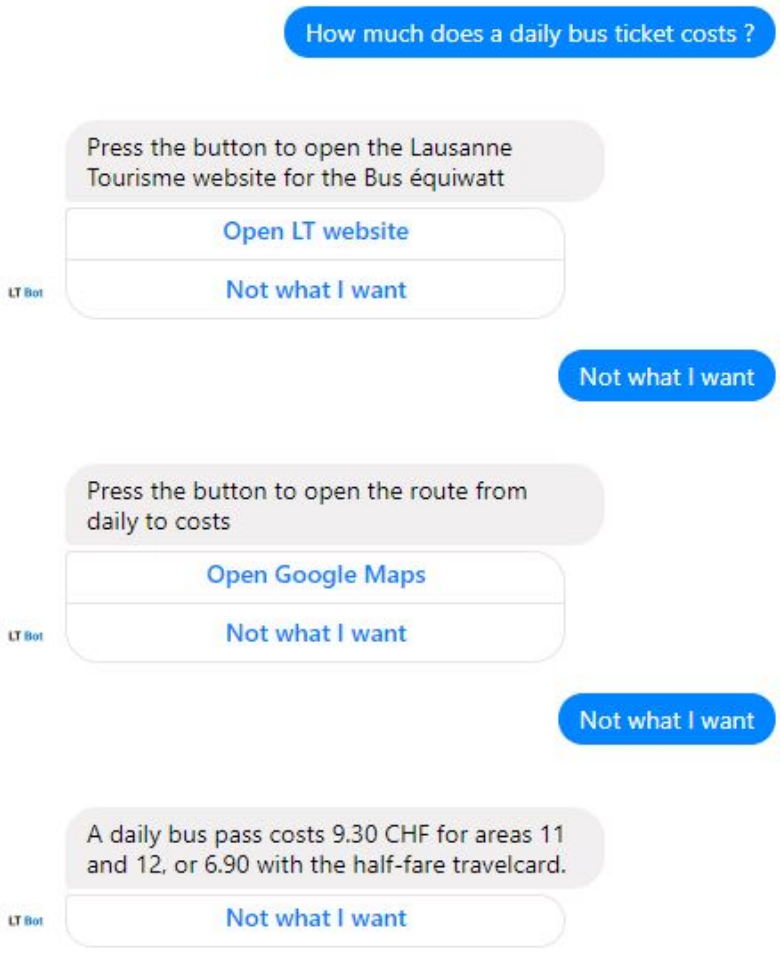

FIGURE 5 – Réponse du chatbot à la 3ème tentative

10.3 Résultats des tests

Les données concernant la satisfaction sont données à la table 1, page 25 et sont visualisées à la figure 10, page 27. On obtient au total un taux de satisfaction de 45% (le total des catégories *Très satisfait* et *Plutôt satisfait*) et moins de 40% de réponses qui ont été jugées très insatisfaisantes (généralement quand le système n'était pas capable de répondre à la question posée).

Très satisfait	8
Plutôt satisfait	12
Plutôt insatisfait	8
Très insatisfait	17

TABLE 1 – Données de satisfaction (sur 45 entrées)

Les données concernant le nombre de tentatives de réponse de la part du système sont données à la table 2, page 26 et sont visualisées à la figure 11, page 28. On voit que le chatbot arrive à trouver une réponse à presque 50% des questions posées, résultat satisfaisant pour une première itération du prototype. On remarque aussi qu'aucun participant n'a essayé de reformuler sa question pour obtenir une réponse en cas d'échec (comme nous le montre les 0 entrées de l'option *Plus*).

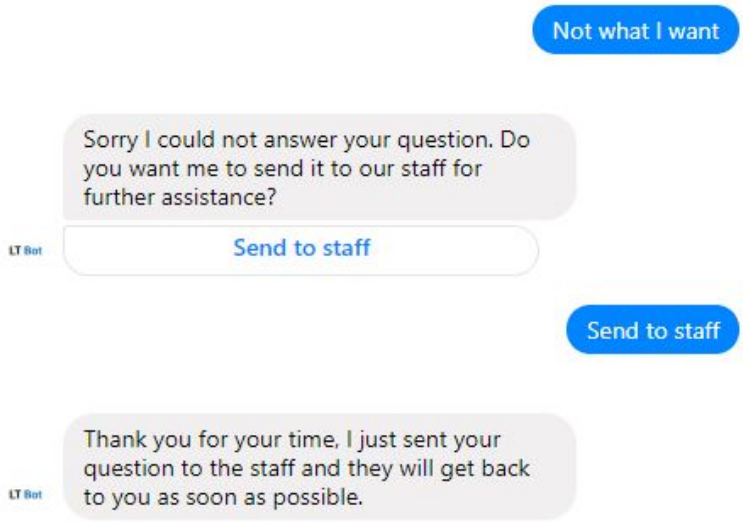

FIGURE 6 – Messages indiquant que le chatbot n'a pas réussi à répondre et demande s'il faut contacter le staff

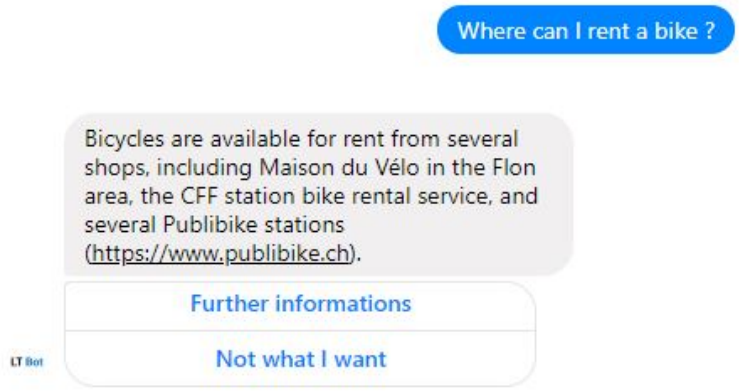

FIGURE 7 – Réponse provenant de la FAQ et contenant un lien hypertexte

1 tour	8
2 tours	12
3 tours	8
Plus	0
Jamais	17

TABLE 2 – Données du nombre de tentatives de réponse (sur 45 entrées)

10.4 Exploration des questions posées

Analysons quelques questions qui ont été posées pour mieux comprendre les chiffres qui ressortent de ce test. Pour chaque catégorie, une petite sélection va être examinée afin d'obtenir plus d'informations sur les points forts et faibles du système. Les questions posées sont notées telles quelles, notamment en gardant l'orthographe, la ponctuation et la casse.

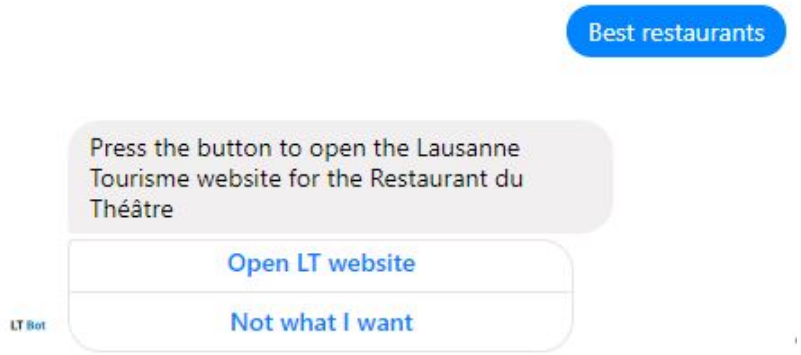

FIGURE 8 – Réponse à une demande de recommandation qui retourne simplement un POI

FIGURE 9 – Réponse à une demande de recommandation qui retourne une liste pertinente provenant de la FAQ

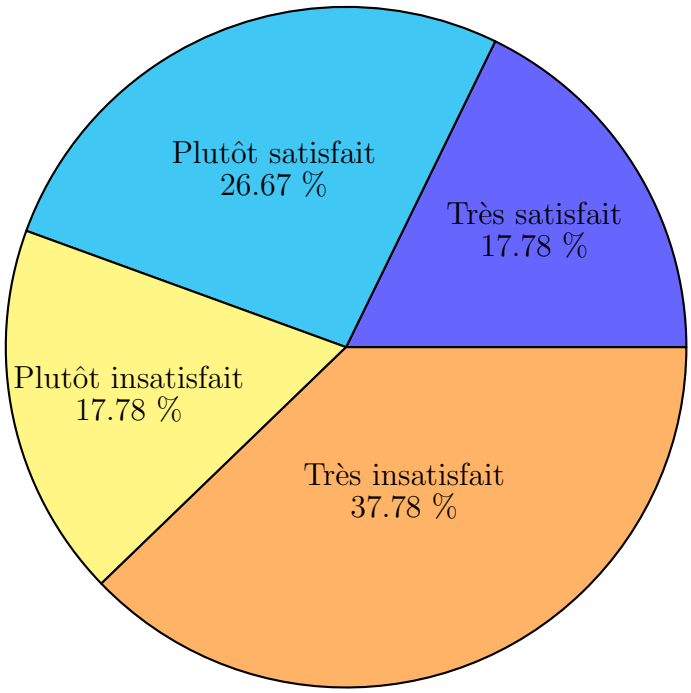

FIGURE 10 – Résultats de la satisfaction des réponses fournies du chabot

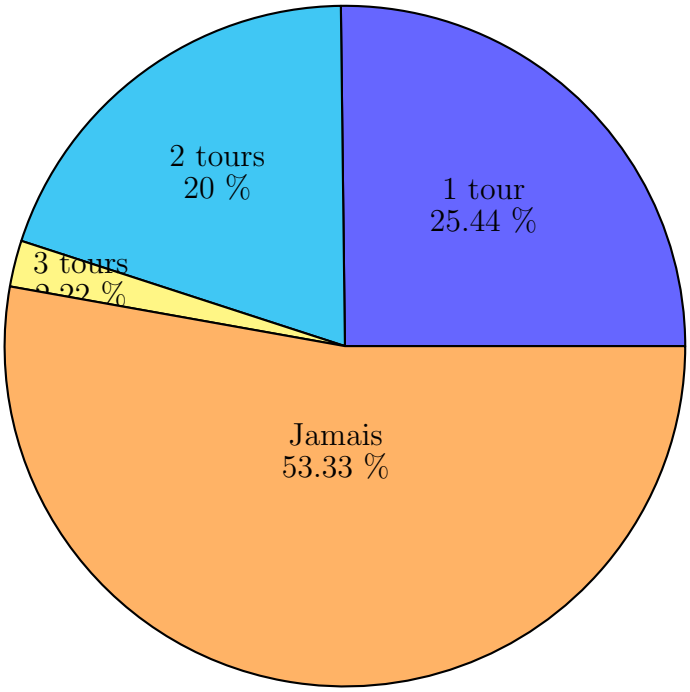

FIGURE 11 – Résultats du nombre de tentatives de réponse du chatbot pour obtenir la bonne réponse

10.4.1 Catégorie très satisfait

Voici quelques questions qui ont été posées dont le chatbot a donné une réponse jugée très satisfaisante par les participants :

- *What are the best places to get a drink ?*, répondu sur le site de Lausanne Tourisme
- *Where can I find a swiss restaurant ?*, répondu par Google Maps
- *where is the lake ?*, répondu par Google Maps
- *What's the events for this week*, répondu avec la FAQ

Cette sélection montre que les trois méthodes de réponse sont capables de répondre à des questions posées et d'obtenir un meilleur score par rapport aux autres en fonction de ce que l'utilisateur entre dans le système. Pour presque tous les résultats de cette catégorie, le chatbot fourni la réponse que l'utilisateur souhaite à la première tentative. La seule exception est la 2ème question qui est répondue à la 2ème tentative.

Généralement les questions qui tombent dans cette catégorie sont des recherches de lieux particuliers. Peu d'entrées de la FAQ ont permis au système de répondre directement à l'utilisateur. La faible diversité des testeurs en est certainement la cause.

10.4.2 Catégorie plutôt satisfait

Voici quelques questions qui ont été posées dont le chatbot a donné une réponse jugée plutôt satisfaisante par les participants :

- *What are the best burgers to eat in town ?*, répondu à peu près par le site de Lausanne Tourisme
- *Where is the coolest park ?*, répondu par Google Maps
- *Where can I buy a souvenir ?*, répondu avec la FAQ
- *How do i go to the lake ?*, répondu par Google Maps

Dans cette catégorie on obtient aussi des résultats avec les trois différentes méthodes de réponse. Le système ne répond pas forcement à toutes les questions, mais fourni souvent une réponse qui contient assez d'éléments pour satisfaire l'utilisateur. La limite entre cette catégorie et *plutôt insatisfait* varie d'un utilisateur à l'autre, car les résultats sont subjectifs quand il y a seulement une partie de la réponse qui est fournie.

10.4.3 Catégorie plutôt insatisfait

Voici quelques questions qui ont été posées dont le chatbot a donné une réponse jugée plutôt insatisfaisante par les participants :

- *Is it possible to have a tour on the lake by boat ?*, répondu environ par le site de Lausanne Tourisme
- *where is the tourist office please ?*, pas répondu
- *How much does a daily bus ticket cost ?*, répondu avec la FAQ
- *what are the hotels near me ?*, répondu environ par le site de Lausanne Tourisme

Les réponses qui se classifient dans cette catégorie sont soit sans lien avec la question posée soit trop vagues pour fournir une réelle information à l'utilisateur, mais arrivent néanmoins à donner quelques fois des pistes de réponse. On retrouve pas mal de questions qui pourraient avoir leur place dans la FAQ (par exemple la 2$^{\text{ème}}$ question sur l'office du tourisme).

10.4.4 Catégorie très insatisfait

Voici quelques questions qui ont été posées dont le chatbot a donné une réponse jugée très insatisfaisante par les participants :

- *How do I get to Zermatt*
- *What is the weather tomorrow*
- *Does the train ticket work with buses ?*
- *How many church do the city count ?*
- *What are the best thing to do in Lausanne*
- *Who is the president of switzerland*

Cette catégorie comporte, sans surprise, toutes les questions qui n'ont jamais été répondues. Mais dans ces questions, certaines sortent du cadre dans lequel le chatbot est sensé pouvoir répondre (comme la 1ère sur Zermatt, la 2ème sur la météo ou encore la 6ème concernant le président de la Suisse).

Certes le système n'a pas pu répondre à ces questions, mais si l'on considère comme normal que celles qui sont en dehors du contexte de la ville de Lausanne n'ai pas pu être répondues, alors on obtient des résultats globaux encore plus satisfaisants.

10.5 Tests automatiques

En plus des tests fait du chatbot, certains tests de validation des algorithmes d'attribution de score ont aussi été fait.

10.5.1 Tests automatiques de la FAQ

Les questions utilisées pour effectuer les tests ont été choisies en fonction du contenu de la FAQ. Pour chacune des catégories, des questions ont été conçues pour être catégorisée dans celles-ci. Voici quelques exemples de questions :

```
"faq": {
  "high": [
    "Is there a hop-on hop-off bus or a bus tour of the town ?",
    "Where are the CFF counters ?",
    "cultural events happening",
    "Christmas market held",
    "underground parking lots",
    "train ride Geneva"
  ],
  "medium": [
    "events happening",
    "market held",
    "underground parking",
    "train ride",
    "Can we do a bus tour all around the town ?",
    "Does the city provide some guided tours ?"
  ],
  "low": [
    "What is the name of the football club of Lausanne ?",
    "What are the most famous clubs ?",
    "Malley",
    "mountain",
    "lake",
    "station"
  ]
}
```

Puis les résultats sont présentés ainsi :

```
question: Where are the CFF counters ?
  result: Where are the CFF counters ?
  score: 7.755547176054053
  result: Where are the currency exchange offices ?
  score: 3.087935313441653
  result: Where are the lockers at the train station ?
  score: 1.3378460300049477
```

```
  result: How far is the center from the station ?
  score: 1.1419774516162742
  result: How far is the lake from the station ?
  score: 1.0493464632808365
  result: Where can I rent a bike ?
  score: 1.0493464632808365
  result: Where can I find duty exemptions forms for purchases made abroad
      ?
  score: 1.0217692988145612
score: 1
```

Pour cet exemple, on voit qu'il y a une entrée dans la FAQ qui est mot pour mot la même que la question. Le premier résultat retourné a donc un score élevé et il se démarque bien des autres résultats. Pour cette question, le résultat est 1, c'est-à-dire la catégorie *Élevé*.

C'est en analysant les résultats que l'algorithme d'attribution de score a pu être affiné. La difficulté étant d'éviter de trop ajuster les paramètres pour que l'algorithme fonctionne bien pour ces exemples, mais ne classifie pas aussi bien d'autres questions (le concepte d'*overfitting*).

10.5.2 Tests automatiques des POIs

Une stratégie similaire a été employée pour les POIs. Comme ils sont notés uniquement par rapport à leur indice de confiance retourné par LunR, il y a moins de place pour ajuster les scores obtenus. On obtient alors des résultats qui sont généralement trop élevé. Voici quelques exemples de questions :

```
"poi": {
  "high": [
    "Lausanne Palace (14 points GM)",
    "Restaurant Tom-Yam",
    "There is no finer setting in the city centre for private functions
       and professional meetings",
    "Thai minced chicken with basil, Massaman curry, soft shell crab salad
       , and lobster in red curry are specialties of this Thai restaurant
       with 13 points in the GaultMillau."
  ],
  "medium": [
    "Cathedral",
    "festival",
    "Marathon"
  ],
  "low": [
    "Zermatt",
    "Lyon",
    "pottery"
  ]
}
```

Puis les résultats sont présentés de la même manière :

```
question: Tom-Yam
  result: Restaurant Tom-Yam
  score: 11.639113985445421
  result: Tom Café
  score: 4.664783435487654
```

```
6     result: Tom Walker (UK)
7     score: 4.1139472529433325
8   score: 1
```

Ici l'exemple est classifié comme on voudrait qu'il le soit. En effet, le premier résultat semble être celui recherché et le reste correspond à peu près à l'entrée (même si cela n'a pas d'influence sur les réponses, car le contrôleur considère uniquement le premier résultat). Par contre, d'autres résultats sont vraiment mal classifié, comme par exemple :

```
1   question: Lyon
2     result: Palais de Rumine
3     score: 2.484
4   score: 1
```

Cette entrée obtient un score *Élevé* alors que l'on aurait voulu qu'elle soit au contraire classifiée dans la catégorie *Faible*.

11 Résultats

Pour certaines questions, le chatbot fonctionne bien. Mais il est inconsistant et n'arrive pas à répondre à d'autres questions similaires. Ou alors il y arrive, mais après plusieurs tentatives (par exemple figure 5, page 25). L'ajustement des algorithmes d'attribution de score de chaque méthode est délicat, car il est difficile de prévoir toutes les interactions possibles.

Lors de la phase de test, beaucoup de POIs étaient proposé en premier lieu en guise de réponse sans être toujours pertinents. La méthode d'attribution de score devrait être améliorée, mais comme il y a beaucoup de POIs (plus d'un millier), LunR va souvent retourner des résultats et donc il est compliqué d'anticiper le degré de confiance retourné.

Lors que le chatbot arrive à répondre correctement à la première tentative (par exemple figure 7, page 26), les testeurs sont toujours agréablement surpris. Ue piste d'amélioration serait d'augmenter l'exigence des algorithmes et opter pour une stratégie plus élitiste que celle mise en place consistant à essayer de répondre à tout prix à la question posée pour obtenir de meilleurs résultats.

Globalement, les performances du chatbot sont plutôt bonnes et pourraient être améliorées encore plus sans trop de changement dans l'architecture (voir section 12), mais simplement en ajustant les différents paramètres du système.

En envisageant des modifications plus profondes (voir section 13) et donc plus coûteuse (voir section 14), un système plus efficace et surtout moins paramétré par rapport aux données de test peut être envisagé. On imagine un tel système plus robuste et plus performant.

12 Améliorations et futurs travaux

Dans cette section serons présentés les améliorations envisageables aux composants actuels du système ainsi que des ajouts possibles dans l'architecture utilisée actuellement. La section suivante présentera une généralisation possible du système dans le but de pouvoir appliquer ce genre d'architecture dans d'autres situations d'utilisation.

12.1 Améliorations possibles du système

Chaque module du système peut être amélioré et affiné pour le rendre le plus pertinent que possible. Néanmoins, les changements effectués rendront effectivement le système meilleur, mais n'apporteront pas une énorme amélioration, car il s'agit simplement d'affiner les paramètres déjà utilisés actuellement.

12.1.1 WIT.AI

WIT.AI a été configuré pour détecter des lieux dans les questions posées par les utilisateurs. Des améliorations sont possibles en ajoutant d'autres entités que le système pourrait comprendre, comme par exemple des trajets ou des horaires. Pour chaque entité ajoutée, il faudrait choisir comment répondre à l'utilisateur. En effet, quand le chabot repère un lieu, il répond avec un liens sur Google Maps. Pour des trajets, on pourrait imaginer la même procédure. Par contre pour des horaires, il faudrait développer une nouvelle manière d'afficher les informations en plus d'entraîner le système à les reconnaître.

12.1.2 Recherche dans la FAQ

Une solution simple pour améliorer la recherche dans la FAQ est d'ajouter des questions/réponses, mais cela demande beaucoup de temps, surtout si l'on insère des données vraiment utiles (par exemple après une enquête auprès des utilisateurs).

12.1.3 Recherche dans les POIs

Pour pouvoir suivre l'évolution des POIs du site de Lausanne Tourisme, il faudrait pouvoir les connecter à la base de données que le site utilise pour les stocker et ainsi faire transparaître toute évolution.

12.1.4 Amélioration de l'algorithme du contrôleur

L'algorithme du contrôleur peut être amélioré en analysant plus précisément des questions posées et la réponse retournée par le système. Comme les scores obtenus ont été attribués en fonction des quelques paires question/réponse, il est possible d'affiner encore mieux les seuils.

De même, les fonctions donnant un score aux différentes méthodes peuvent aussi être affinées. Il est aussi envisageable d'ajouter d'autres classes (en plus que faible, moyen et élevé) de confiance pour permettre plus de nuances. Par contre, l'ajout de classes supplémentaires augmente la complexité de l'algorithme et demande d'ajouter plus de seuils à la main et risque au final de ne pas être assez pertinents.

Pour le moment, le contrôleur prend en compte uniquement le premier résultat retourné par chaque méthode de réponse (pour être exact, la FAQ utilise le 2ème et le 3ème résultat pour calculer le score, mais retourne uniquement le 1er résultat en cas de succès). En modifiant la logique du contrôleur, il serait possible d'inclure plusieurs tentatives de réponse d'une même méthode. Amélioration intéressante dans l'idée, mais qui augmenterait beaucoup la complexité de l'attribution de score de chaque méthode ainsi que la logique du contrôleur.

12.2 Futurs travaux sur l'architecture actuelle

Certaines fonctionnalités n'ont pas pu être implémentées et offriraient une valeur ajoutée au chatbot. Les mettre en place ne demanderait pas énormément de travail et offrirait une utilisation plus agréable. Par contre il ne s'agit pas de fonctionnalités permettant d'accroire les performances du système, mais elles rendrait l'utilisation de l'agent plus agréable.

En voici la liste :

- Enseigner à *WIT.AI* à reconnaître des entités supplémentaires (autre que celles de base)
- Effectuer une recherche sur le site de Lausanne Tourisme pour y extraire le meilleur résultat (POI) de manière dynamique
- Contextualiser chaque interaction par rapport à la géolocalisation de l'utilisateur
- Insérer des images dans les réponses formatées pour afficher les réponses des POI
- Améliorer le système au cours du temps en stockant les questions les plus fréquemment posées et en les ajoutant à la FAQ

13 Généralisation

Dans *Data Mining : Practical machine learning tools and techniques* [11], au chapitre 3, Witten et al. exposent plusieurs moyens pour décrire des données et les relations entre leur différentes classifications. De telles méthodes pourraient être utilisées pour inverser la logique du contrôleur. Au lieu de partir des différentes combinaisons de scores de chaque méthode pour choisir alors quel méthode on choisirait (par exemple si on a la FAQ et les POIs qui retournent une réponse considérée élevé et WIT.AI retourne un score faible, on définit que le contrôleur considère la réponse de la FAQ comme étant la meilleure), on fait l'inverse, en mettant en place un apprentissage automatique de la manière suivante :

- Mettre en place une liste de questions dont la réponse devrait se trouver dans les différentes méthodes avec une distribution la plus uniforme possible
- Garder les scores brutes de chaque méthode de réponse à chaque question
- Labéliser à la main chaque question en indiquant la meilleure réponse
- Entraîner un réseau de neurones sur ces données labellisées pour ainsi obtenir un modèle capable de retourner la meilleure méthode de réponse

Cette approche souffre du fait que pour obtenir un réseau efficace, il faut avoir un maximum de données sur lequel l'entrainer et donc demande beaucoup de travail au préalable pour avoir un système efficace. On perd aussi en modularité (ce qui n'est pas forcément une mauvaise conséquence ; on n'en a pas forcement besoin). Avec cette technique, ajouter une nouvelle source de savoir demanderait de re-entraîner le réseau pour inclure dans son algorithme cette nouvelle possibilité de réponse.

Par contre, on se libère d'une attribution de score presque aléatoire, fixée en fonction de quelques paires question/réponse. En gardant les scores bruts retournés par chaque méthode de réponse, on n'a plus besoin de définir des seuils à la main. On obtient alors un algorithme qui est plus proche de ce qu'il se passe derrière la façade lors de l'analyse des différents modules et donc qui ne semble pas sortir de nulle part. De plus, en utilisant beaucoup de données pour entraîner le réseau, le système sera performant face aux nouvelles questions posées.

En imaginant suffisamment de ressources attribuées, on peut même partager les données en un set pour l'apprentissage (mise en place des différents paramètre du réseau), éventuellement un set de validation pour ajuster ces paramètres et finalement un set de test pour juger sa performance sur des données nouvelles non-utilisées lors de la mise en place du réseau de neurones. Chaque question devra avoir été préalablement dotée d'un label indiquant quel est la meilleure méthode pour y répondre pour permettre tous ce processus.

Ainsi, on pourrait extraire une analyse plus rigoureuse du système en utilisant des scores utilisés de manière générale pour ce genre d'évaluation :

- **Précision$_i$** =

$$\frac{\text{nombre de questions correctement attribués à la méthode de réponse } i}{\text{nombre de questions attribués à la classe } i}$$

- **Rappel$_i$** =

$$\frac{\text{nombre de questions correctement attribués à la méthode de réponse } i}{\text{nombre de questions appartenant à la méthode de réponse } i}$$

- **F1-score** =

$$\frac{2}{(\frac{1}{précision} + \frac{1}{rappel})}$$

Plus le F1-score se rapproche de 1, plus le système est performant. Pour cela, il faut que la précision et le rappel se rapprochent aussi de 1, les deux conjointement.

14 Perspectives

Plusieurs approches sont possibles pour la suite de ce projet :

1. Amélioration de l'architecture actuelle
2. Simplification de l'architecture actuelle
3. Changement d'architecture

Chacune des approches offre un compromis entre performances et coût de mise en place, et a potentiellement son utilité en fonction des objectifs recherchés.

14.1 Amélioration de l'architecture actuelle

Cette première approche consiste à implémenter un maximum d'améliorations proposées en section 12. En affinant les paramètres des fonctions d'attribution de score aux différentes méthodes de réponse, on pourrait augmenter les performances sans changement de logique dans le code, donc de manière aisée.

En récoltant encore plus d'exemples réels d'interaction de touriste venant visiter Lausanne, on pourrait aussi améliorer ces paramètres ainsi que la FAQ pour les questions récurrentes. Les autres propositions de travaux futurs (ajouter des images aux réponses provenant des POIs, contextualiser les interaction, etc.) amélioraient aussi l'utilisation du chatbot et le rendrait plus attrayant.

On peut aussi incorporer une nouvelle source de savoir (en effet, l'architecture actuelle est modulable et l'ajout d'un nouveau module est simple) pour cibler certaines réponses. Par exemple connecter le chatbot aux horaires des transports lausannois et/ou des trains.

Tous ces changements, individuellement, ne demanderaient pas beaucoup d'efforts et augmenteraient de manière incrémentale les performances sans changement drastique. Néanmoins, plus les ajouts de fonctionnalités sont effectués, plus la complexité de l'application augmente, avec ses risques d'erreurs.

On a là une approche qui ajouterai de petites performances à coût réduit, proportionnellement au temps investi à l'implémentation des différentes améliorations, jusqu'à une certaine limite où l'ajout de valeur ajoutée deviendrait risible par rapport au temps consacré.

14.2 Simplification de l'architecture actuelle

Cette deuxième approche consiste, à l'instar de la précédente, à simplifier le système actuel en modifiant les objectifs du chatbot. On ne cherche donc plus à accomplir les mêmes tâches, mais un sous-ensemble de celles-ci afin de fortifier les capacités de l'application.

Cette stratégie provient des observations faites lors de la phase de test. En effet, quand le chatbot arrivait à répondre immédiatement à la question, les réactions étaient toutes positives. Par contre quand il essayait de répondre avec une réponse qui n'avait aucun rapport avec la question posée, là les réactions étaient négatives. Certaines questions étaient automatiquement jugées comme trop difficile à répondre pour un bot et certains ne s'étonnaient pas que le système avait été mis en échec. Le contraste entre la compréhension que la question était délicate à répondre et toutes les tentatives malgré tout fournies à l'utilisateur met en lumière une possibilité d'augmenter les exigences concernant les tentatives de réponse.

Ainsi, en réduisant de manière drastique la classification des réponse fournies (par exemple en enlevant la classe "Moyen" et gardant qu'une classe jugée comme répondant à la question et une autre jugée comme ne répondant pas à la question) et en n'hésitant pas à indiquer que le chatbot n'arrive pas à répondre, l'on obtiendrait une expérience qui surprendrait en bien lors d'une réponse (qui sera par définition de cette nouvelle approche plus enclin à être une bonne réponse) et procurant un avis plutôt neutre lors d'un échec de réponse.

Pour aller à l'extrême de cette approche, toutes les sources de savoir, à l'exception de la FAQ, seraient supprimées. Comme la FAQ comporte des paires question/réponse dont la question a effectivement été posée par des touristes, on obtiendrait une base de savoir pertinente. De plus, si une question posée par un utilisateur est similaire à une entrée de la FAQ, on est sûr que la réponse associée répondra efficacement à sa question.

L'effort nécessaire à mettre en place est minime. Il faudrait modifier le code pour enlever toutes les parties alors inutiles et se concentrer à définir le seuil jugeant à quel niveau de similarité une entrée de la FAQ est jugée similaire à la question posée.

Cette approche a l'avantage de coûter très peu et sera rapidement mise en place. Par contre on diminue les performances du chatbot, car on lui enlève des capacités de réponse. Il s'agit donc d'un compromis entre essayer de répondre à un maximum de requête versus admettre que le chatbot ne puisse pas répondre à tout et dès lors transmettre directement la question au staff.

14.3 Changement d'architecture

Cette dernière approche consiste à mettre en place les modifications décrites en section 13 et donc introduire de nouvelles techniques d'apprentissage automatique pour améliorer les performances du système.

Il s'agit de l'approche qui demande le plus de travail, car il faut presque tout recommencer pour reconstruire toute la logique de l'application et prendre en compte le réseau de neurones. Mais le plus long est de mettre en place les données labellisées qui seront utilisées pour l'apprentissage, la validation puis le test du réseau de neurones.

Par contre une fois le réseau mis en place (en partant du principe qu'assez de données ont été utilisées pour l'entraînement), il sera plus performant et surtout plus robuste face aux questions inattendues, car il aura défini des stratégies basées sur la structure des questions posées.

Plus complexe à mettre en place, la logique du fonctionnement devient ensuite plus simple à comprendre. En effet, la logique du contrôleur revient simplement à entrer la question et les résultats de méthode de réponse dans le réseau et il s'occupe de ressortir ce qu'il estime être la meilleure réponse. Comme tout réseau de neurones, le fonctionnement exacte en interne est plus obscure et demande la compréhension de ce genre d'algorithme pour le comprendre dans sa totalité.

L'effort nécessaire à mettre en place cette approche est le plus élevé. Il faut à peu près refaire complètement le chatbot et en plus il faut mettre en place les données labellisées ainsi que le réseau de neurones.

L'intérêt de cette approche est qu'il s'agit de celle qui a le plus de potentiel d'obtenir des performances élevées. On a donc un coût élevé pour de grandes performances. Cette stratégie est la plus désirable si l'on veut maximiser les capacités du chatbot et qu'on a les moyens à disposition.

15 Conclusion

En l'état, le chatbot fournis des résultats satisfaisants grâce aux différentes méthodes employées pour répondre aux questions posées par les utilisateurs. L'analyse par WIT.AI permet de répondre aux questions cherchant un lieu ou aux formules de politesse. La FAQ sert pour les questions précises et techniques. Puis les POIs permettent de répondre à tous les éléments d'intêret de la ville de Lausanne. En combinant ces trois méthodes de réponse, le système arrive à (presque) toujours renvoyer un élément de réponse à l'utilisateur.

Le contrôleur est modulable et offre donc place à l'ajout aisé d'autres sources de savoir dans le système. Sa logique est simple et ses performances dépendent de la qualité des systèmes d'attribution de score des différentes méthodes de réponse. Lors de la phase de test, le chatbot a répondu avec beaucoup de faux positifs en retournant un POI qui ne répondait malheureusement pas à la question posée. Des améliorations concernant cette méthode permettraient d'obtenir rapidement de meilleurs résultats.

Plusieurs autres pistes d'amélioration du système ont été explorées. Globalement les capacités du système peuvent toutes être affinées en investissant un petit effort et fourniraient de meilleures réponses rapidement. L'ajout de quelques fonctionnalités améliorant l'utilisation permettraient de rendre plus agréable l'interaction avec le chatbot en demandant un temps raisonnable d'implémentation.

Un changement d'architecture et de logique de l'application a été entrevu avec l'exploration de la mise en place d'un réseau de neurones permettant de choisir lui-même les seuils définissant les différentes catégories de qualité de la méthode de réponse. Demandant une plus grande quantité de travail en amont, cette solution permettrait de ne pas avoir à modifier à la main (et sans trop de retour concret) les différents paramètres des systèmes d'attribution de score des différentes méthodes.

Finalement une analyse d'approches possibles pour la suite du projet ont été analysées avec chacune des objectifs et compromis coût/performance différents. Personnellement, je trouve les trois approches intéressantes. Mais au vu des performances actuelles (plutôt bonnes) du

chatbot, je me dirigerai plutôt sur la mise en place d'une nouvelle architecture en mettant en place une approche d'apprentissage automatique.

D'un autre côté, si le budget n'est pas présent, je trouverai intéressant de simplifier les objectifs recherchés à travers ce chatbot pour ainsi pouvoir augmenter la qualité des réponses fournies et plus rapidement annoncer à l'utilisateur qu'aucune réponse n'a été trouvée. Bien sûr, cette solution n'est pas très attrayante, car il s'agit finalement d'une simple FAQ interactive, mais elle a néanmoins sa place dans les possibilités d'évolution du projet.

Pour conclure, ce projet m'a permis d'envisager des solutions nouvelles concernant la gestion des différentes méthodes de réponse aux questions posées. Encore beaucoup d'améliorations sont possibles et plusieurs moyens pour y parvenir sont envisageables. Le produit fournis permet de se rendre compte de la difficulté à mettre en place une solution efficace et performante. Au vu de dernier progrès des dernières années, ce domaine a encore beaucoup de place pour de l'innovation, que ce soit en appliquant une seule stratégie ou en combinant plusieurs d'entre-elles.

Et pour laisser le dernier mot au chatbot :

Références

[1] Sanda M. Harabagiu, Marius A. Paşca, and Steven J. Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 292–298. Association for Computational Linguistics, 2000.

[2] Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2) :133–154, 2003.

[3] Pum-Mo Ryu, Myung-Gil Jang, and Hyun-Ki Kim. Open domain question answering using wikipedia-based knowledge model. *Information Processing & Management*, 50(5) :683–692, 2014.

[4] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv :1506.05869*, 2015.

[5] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784, 2016.

[6] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv :1704.00051*, 2017.

[7] Rafael E Banchs and Haizhou Li. Iris : a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations*, pages 37–42. Association for Computational Linguistics, 2012.

[8] Zongcheng Ji, Zhengdong Lu, and Hang Li. An information retrieval approach to short text conversation. *arXiv preprint arXiv :1408.6988*, 2014.

[9] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv :1503.02364*, 2015.

[10] Marius Paşca. Open-domain question answering from large text collections, 2003.

[11] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining : Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

Annexes

A Cahier des charges initial

1. Spécification du domaine dans lequel le chatbot sera opérationnel : brève analyse des concepts du domaine, collection d'exemples de requêtes. Accès au contenu des sites web Lausanne tourisme et MyLausanne.

2. Examen comparatif de différentes plateformes permettant l'implémentation de chatbots. Sélection d'une solution technologique. Choix de la langue du prototype (français, anglais ou multilingue).

3. Création/adaptation d'une interface et d'un modèle d'interaction pour le contexte d'un guichet touristique. Gestion de la fin d'interaction dans les cas où l'intervention d'un opérateur humain est requise.

4. Création du système de réponses aux question des utilisateurs, en deux étapes-clé [15] : détermination de la page web ou du document susceptible de contenir la réponse, et calcul de la réponse courte proposée.

5. Évaluation du système obtenu avec un petit nombre d'utilisateurs.

6. Rédaction du rapport, incluant des recommandations pour la réalisation d'un système susceptible d'être utilisé en production.

15. Ces deux étapes-clé ont été modifié ainsi : intégration de 3 sources de connaissance (MAP, FAQ et POI) avec pour chacune la formulation de requêtes, puis l'intégration des résultats.

B Installation du chatbot

L'installation du chatbot a été effectuée en suivant la procédure officielle disponible sur le site de Facebook for developers [16].

Plusieurs option sont disponibles :

1. Créer le projet soi-même
2. Télécharger un projet exemple depuis Github
3. Remanier un projet exemple hébergé directement sur Glitch

Pour ce projet, la 3ème option a été choisie et est la plus simple. Sinon, comme il s'agit d'une application `Node.js`, n'importe quelle procédure standard d'exécution fonctionnera, à condition de modifier toutes variables d'environnement permettant les connections entre Facebook et l'application.

Pour une exécution correcte, il faut modifier ces variables dans le fichier `.env` sur Glitch :

```
1 PAGE_ACCESS_TOKEN="<EAA...>"
2 VERIFY_TOKEN=<"9XG...>"
3 USER="<...@ethereal.email>"
4 PASS="<...>"
```

C Exécution du crawler

Le crawler est une application `Node.js` aussi. Pour l'exécuter, il faut lancer les commandes suivantes :

```
1 npm install
2 npm start
```

16. https://developers.facebook.com/docs/messenger-platform/getting-started/quick-start

D Résultats des tests

Personne	Question	Satisfaction	Tours	Remarques
O.M.	Where is the lake	Plutôt satisfait	2	maps
O.M.	How do I get to Zermatt	Très insatisfait	jamais	
O.M.	What is the weather tomorrow	Très insatisfait	jamais	
N.T.	Is it possible to have a tour on the lake by boat ?	Plutôt insatisfait	jamais	LT environ
N.T.	Can I buy a one-day ticket for all public transports ?	Très satisfait	1	LT
N.T.	Are all museums open on mondays ?	Très insatisfait	jamais	
R.B.	What are the main places to party here in Lausanne ?	Plutôt insatisfait	jamais	LT environ
R.B.	What are the best burgers to eat in town ?	Plutôt satisfait	jamais	LT environ
R.B.	Where can I eat fondue/ raclette/ swiss dishes ?	Plutôt insatisfait	jamais	
E.B.	Where can I find the best beer in Lausanne ?	Plutôt satisfait	1	LT
E.B.	When was the cathedral built ?	Plutôt satisfait	1	LT
A.D.W	Where is the best rooftop ?	Plutôt satisfait	2	maps environ
A.D.W	Where is the coolest park ?	Plutôt satisfait	1	maps
A.D.W	Does the train ticket work with buses ?	Très insatisfait	jamais	
G.T.	Where is the train station ?	Plutôt satisfait	2	maps
G.T.	How do we buy a metro ticket ?	Très insatisfait	jamais	
G.T.	Can we walk to get to the Flon or should we take a taxi/uber ?	Très insatisfait	jamais	
A.B.G	What are the best places to get a drink ?	Très satisfait	1	LT
A.B.G	Where is the Cathedral ?	Très satisfait	1	LT + maps
A.B.G	How do I get to Flon from the train station ?	Très insatisfait	jamais	
F.Q.	where is the tourist office please ?	Très insatisfait	jamais	
F.Q.	what are the hotels near me ?	Plutôt insatisfait	1	LT
G.M.	Is it possible to climb on the cathedral's roof ?	Très insatisfait	jamais	
G.M.	How much does a daily bus ticket cost ?	Plutôt insatisfait	3	FAQ
G.M.	When is the Olympic museum opened ?	Très insatisfait	jamais	
Q.G.	How do i go to the lake ?	Plutôt satisfait	2	maps
Q.G.	Why is it so pentu ?	Très insatisfait	jamais	
Q.G.	Where is the olympic museum	Plutôt satisfait	2	maps
C.	Where can I withdraw/convert money ?	Très insatisfait	jamais	
C.	Where is the train station ?	Plutôt satisfait	2	maps
C.	Where is the olympic museum ?	Très satisfait	1	LT + maps

Personne	Question	Satisfaction	Tours	Remarques
C.L.	Where is the lake ?	Très satisfait	1	maps
C.L.	Where can I buy a souvenir ?	Plutôt satisfait	2	FAQ
C.L.	Where can I find a swiss restaurant ?	Très satisfait	2	maps
R.J.	where is the Olympic museum	Très satisfait	1	LT
R.J.	what are the best clubs to go	Très insatisfait	jamais	
R.J.	how can I go from the hotel to the Olympic museum	Plutôt satisfait	2	maps
M.V.	What the see level difference when you go from the lake to the top part of the city ?	Très insatisfait	jamais	
M.V.	How many church do the city count ?	Très insatisfait	jamais	
M.V.	Why is there a man sleeping on the top of the cathedral ?	Très insatisfait	jamais	Lt des infos sur la cathedral
T.C.	What are the best thing to do in Lausanne	Très insatisfait	jamais	
T.C.	What's the events for this week	Très satisfait	1	FAQ
T.C.	Who is the president of switzerland	Très insatisfait	jamais	

Authentification

Par la présente, je soussigné, Gabriel Luthier, déclare avoir réalisé seul ce travail et ne pas avoir utilisé d'autres sources que celles citées dans les références.

Date

Signature

Nom complet