Projet TATIA : Rapport :

Conception et implémentation d’un système de questions réponses en langue naturelle sur des données structurées avec système de reconnaissance vocale intégré

Les systèmes de questions/réponses sont des systèmes qui répondent à une question posée en langage naturel, par l’extraction d’une réponse précise à partir d’un corpus de documents. Le projet que nous avons réalisé ici s’appuie sur base *DBpedia* qui propose une version structurée et normalisée au format du web sémantique des contenus de *Wikipedia*. Le système que nous avons conçu requiert l’utilisation d’un **interpréteur Python** afin de permettre son exécution ainsi que d’une **connexion internet active** afin d’extraire les réponses de la base *DBpedia*. Le programme a en particulier été testé avec Python en **version 3.9 et 3.9.1** mais devrait cependant fonctionner sous des versions plus anciennes ou plus récentes de **Python 3**. De plus un certain nombre de packages Python supplémentaires doivent être installés sur la machine afin de permettre l’exécution du programme. Le nom des packages et les différentes commandes d’installation sont détaillés dans la première section de ce rapport.

Table des matières

[Prérequis au lancement du programme](#_Toc60282073)

[Packages indispensables pour le fonctionnement du projet](#_Toc60282074)

[L’installation de tkinter est nécessaire à l’affichage de l’interface graphique (gui)](#_Toc60282075)

[Packages nécessaires aux fonctionnalités vocales](#_Toc60282076)

[Note concernant l’installation de PyAudio](#_Toc60282077)

[Choix de Spacy](#_Toc60282078)

[Fonctionnement global du traitement question / réponse](#_Toc60282079)

[Configuration du projet](#_Toc60282080)

[Détail du code par fonctions](#_Toc60282081)

[Type de questions prises en compte dans notre programme](#_Toc60282082)

[Interface graphique (gui)](#_Toc60282083)

[Fonctionnalités vocales](#_Toc60282084)

[Reconnaissance vocale](#_Toc60282085)

[Lecture vocale](#_Toc60282086)

[Evaluation de notre système](#_Toc60282087)

[Améliorations possibles](#_Toc60282088)

# **Prérequis au lancement du programme**

En plus **d’un interpréteur Python et d’une connexion internet active,** des packages Python supplémentaires sont nécessaires au fonctionnement du code de ce projet. La liste ci-après contient le nom des packages à installer ainsi que la commande permettant son installation avec le gestionnaire de paquet ***PIP*** accompagné d’une brève description (nous reviendrons en détail sur le rôle concret des éléments apportés par ces packages dans notre projet plus tard dans ce rapport).

Les packages peuvent être installés séparément ou d’un seul coup avec les 2 lignes suivantes :

pip install spacy bs4 lxml deep-translator SpeechRecognition gtts pyglet PyAudio

+ python -m spacy download fr\_core\_news\_lg

## Packages indispensables pour le fonctionnement du projet

* [**Spacy** :](https://spacy.io/usage) pip install spacy

*SpaCy est une bibliothèque logicielle Python libre de traitement automatique des langues. Cette bibliothèque inclut par exemple des outils tels qu’un tokenizer ou un PoS tagger, mais a aussi pour grand avantage d’inclure un NER prenant en charge la langue française.*

* [***Modèles statistiques pré-entraînés en français pour Spacy :***](https://spacy.io/models/fr)

python -m spacy download fr\_core\_news\_lg ***(après installation de spacy!)***

*Installation d’un modèle français pré-entraînés pour Spacy permettant la prédiction des entités nommée et la détermination des dépendances syntaxiques. Il s’agit ici d’un réseau neuronal convolutif formé sur UD French Sequoia et WikiNER en français.*

***Note : un bug interne à numpy version 1.19.4 peut faire planter, dans ce cas revenir à la version 1.19.3 :*** pip uninstall numpy ***et*** pip install numpy==1.19.3

* [***Beautiful Soup 4:***](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)pip install bs4

*Bibliothèque Python permettant d'extraire des données de fichiers HTML et XML.*

* [***Lxml:***](https://lxml.de/installation.html)pip install lxml

*Bibliothèque permettant le traitement de XML et HTML dans le langage Python.*

* [***Deep Translator:***](https://pypi.org/project/deep-translator/)pip install deep-translator

*Bibliothèque permettant la traduction entre différentes langues de manière simple en utilisant plusieurs traducteurs.*

## L’installation de tkinter est nécessaire à l’affichage de l’interface graphique (gui)

[***Tkinter***](https://docs.python.org/3/library/tkinter.html)*(de l'anglais Tool kit interface) est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques. Elle vient d'une adaptation de la bibliothèque graphique Tk écrite pour Tcl*

Note : le package est normalement inclus nativement avec Python mais si ce n’est pas le cas, exécuter, si nécessaire, sudo apt-get install python3-tk sous Linux / Ubuntu.

## Packages nécessaires aux fonctionnalités vocales

* [***Speech Recognition:***](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)pip install SpeechRecognition

*Permet la transformation de phrases orales en texte écrit*

* [***gTTS:***](https://lxml.de/installation.html)pip install gtts

*Permet la transformation de texte écrit en fichier mp3*

* [***Pyglet:***](https://pypi.org/project/deep-translator/)pip install pyglet

*Permet la lecture de fichier mp3*

* [***PyAudio:***](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)pip install PyAudio sudo apt-get install python-tk

*Outils assurant divers traitements audios nécessaires au fonctionnement de certaines fonctionnalités de reconnaissance vocale du projet.*

## Note concernant l’installation de PyAudio

Dans certains cas l’installation de PyAudio avec la commande *« pip »* peut s’avérer capricieuse, en particulier avec des versions récentes de Python et/ou de la commande *pip*. On notera ici deux solutions qui devraient permettre la résolution de ce problème selon le système d’exploitation utilisé.

Avec **Windows** les deux commandes suivantes devraient permettre la bonne installation de PyAudio : pip install pipwin suivi de pipwin install pyaudio

Sous Linux (**Ubuntu)** les deux commandes suivantes devraient permettre la bonne installation de PyAudio : sudo apt-get install -y portaudio19-dev suivi de pip install PyAudio.

Si une erreur persite executer sudo apt-get install python3.9-dev et réessayer d’installer PyAudio.

*Note : Il est possible d’exécuter le programme sans bénéficier des fonctionnalités liées à la reconnaissance ou la lecture vocale. L’installation des packages cités dans cette dernière partie ne sont alors pas nécessaires. Se reporter à la section «****configuration du projet****» plus loin dans ce rapport.*

# **Choix de Spacy**

Le système de questions/réponses que nous avons conçu s’appuie principalement sur les outils apportés par la bibliothèque **Spacy**. Nous avons effectué le choix de Spacy pour plusieurs raisons. Tout d’abord nous souhaitions concevoir notre système en français et Spacy possède des modèles statistiques dans cette langue ainsi qu’un support natif du *Named-Entities Recognition (****NER****)* pour le **français** ce qui n’est pas le cas de ces principaux concurrents comme NLTK. De plus les algorithmes de tokenization utilisées dans la bibliothèque Spacy sont efficaces et rapides et la bibliothèque utilise une approche orientée objet facilement lisible. De plus certaines fonctionnalités telles qu’un support de la méthode d’apprentissage « *word embedding* » sont présentes avec Spacy alors qu’elles ne sont par exemple pas supportées par NLTK.

Pour permettre de meilleurs performances, nous avons fait le choix d’exécuter Spacy sur le GPU de la machine lorsque cela été possible (*spacy.prefer\_gpu()*) et d’utiliser le corpus de texte en français pré-entrainés proposé par Spacy : « *fr\_core\_news\_lg* ».

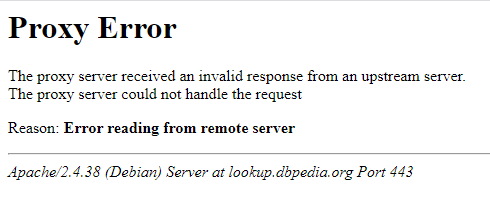
# **Fonctionnement global du traitement question / réponse**

**Cette introduction a pour but d’expliciter de manière générale le fonctionnement du programme que nous avons conçu avant d’entrer plus en détail dans le rôle de chaque fonction. Le rôle général du système de question / réponse que nous avons conçue consiste à prendre une question saisie par l’utilisateur en entrée et à afficher une réponse à cette question en sortie.**

**La 1ère étape consiste donc à extraire de la question posée les éléments qui peuvent nous permettre d’effectuer une requête capable d’afficher la réponse.** Pour cela nous nous sommes principalement appuyés sur des méthodes découlant d’une analyse lexicale (**tokenization**), d’un étiquetage morpho-syntaxique (**POS tagging** [part-of-speech tagging]) et sur une reconnaissance d'entités nommées (**NER** [named-entity recognition]). Ces différentes étapes nous permettent, par exemple, d’extraire les mots-clés les plus importants dans la question (fonction **get\_hotwords**). Afin d’identifier le type de questions posées et de récupérer les éléments nécessaires pour effectuer notre requête nous nous sommes également appuyés sur l’utilisation **d’expressions régulières associés à l’analyse NER** de la question (**fonction exp\_reg**). Cette étape nous permet d’identifier le type de recherche induite par la question (personne, lieu, date, etc) et de conserver les seuls mots nécessaires à l’exécution de la requête.

**La 2ème étape consiste à identifier la page dbpedia sur laquelle se trouve la réponse que l’on recherche.** Afin de pouvoir identifier précisément le nom de la page recherchée on se base sur le mot-clé principal identifié lors de l’étape précédente dans la question et on utilise l’API **DBpedia Lookup** qui nous renvoie le label précis de la page dbpedia correspondant à ce mot-clé (fonction **lookup\_keyword**). L’API nous permet ainsi d’effectuer une requête http renvoyant une page xml associé au mot-clé recherché et grâce à la bibliothèque *BeautifulSoup4* et au parser inclus dans la bibliothèque *Lxml*, on récupère le label exact nous permettant d’accéder à la page dbpedia contenant l’information recherchée. L’API fonctionnant en anglais, on utilise une étape de traduction intermédiaire sur le mot-clé de la question (français vers anglais).

**Note importante : *Nous avons malheureusement fait face à un problème inattendu à cette étape ayant fortement impacté le projet sur lequel nous travaillons. En effet nous utilisions depuis le début l’API suivante :*** [***https://lookup.dbpedia.org/api/search/KeywordSearch***](https://lookup.dbpedia.org/api/search/KeywordSearch) ***(***[***https://github.com/dbpedia/lookup***](https://github.com/dbpedia/lookup)***) qui semble avoir rencontré de forts dysfonctionnements à partir de fin décembre la rendant inutilisable.***



***En conséquence nous avons décidé de changer l’API utilisée :***

[***http://akswnc7.informatik.uni-leipzig.de/lookup/api/search***](http://akswnc7.informatik.uni-leipzig.de/lookup/api/search) ***(***[***https://github.com/dbpedia/dbpedia-lookup***](https://github.com/dbpedia/dbpedia-lookup)***)***

***Le programme continue avec cette nouvelle API de fonctionner globalement de la façon dont nous l’avions conçu mais cela a impacté la dernière ligne droite du projet ainsi que quelques recherches par mot-clé lors de cette étape qui s’effectuaient correctement avec l’API initialement utilisée mais pas avec celle-ci (par exemple le mot-clé « Marseille » de type ville renvoyé bien le label correspondant à la page dbpedia de la ville sur la première API mais pas sur la nouvelle, nous avons cependant implémenté de nouvelles solutions pour régler ou contourner ces problèmes dès que nous en avons constaté).***

**La 3ème étape consiste à exécuter une requête SPARQL sur la page dbpedia identifiée afin de récupérer la donnée correspondant à la réponse à la question.** Pour effectuer cette requête on s’appuie sur la page dbpedia identifiée à l’étape précédente ainsi qu’aux autres mots-clés que l’on avait extrait de la question et qui nous permettent de savoir quelles données extraire de la page dbpedia. La requête elle-même est effectuée en se servant de la fonction **« query »** directement via [l’interface dbpedia](https://dbpedia.org/sparql). La formulation de la requête est effectuée en se servant des mots-clés et de la page précédemment identifiée par les fonctions « **requete\_dbpedia** » ou « **requete\_dbpedia\_multiple** » selon le type de réponse attendue (une seule réponse ou plusieurs réponses [par exemple si on cherche des créateurs pour un site web il peut y en avoir plusieurs mais si on cherche le roi d’un pays il n’y en a qu’un, cela dépend donc du type de la question]).

**Enfin la dernière étape consiste simplement à renvoyer le résultat de la requête qui devrait répondre à la question posée initialement par l’utilisateur.**

# **Configuration du projet**

Il est laissé à l’utilisateur du code de ce projet la possibilité d’effectuer quelques paramétrages afin d’activer ou de désactiver certaines fonctionnalités. Pour cela l’utilisateur doit se reporter à la fonction « **main** » du code contenu dans le fichier Python (à la fin) et configurer les différents booléens préconfigurés soit sur « **True** » soit sur « **False** » selon la configuration désirée. La liste des différents booléens configurables et des conséquences associées est la suivante :

**gui : *True*** pour activer **l’interface graphique*, False*** pour afficher les entrées et sorties dans **la** ***console*** de façon purement **textuelle** sans mise en forme

**vocal : *True*** pour activer **les fonctionnalités de reconnaissance et de lecture vocale*, False*** pour désactiver toutes fonctionnalités vocales (*Note : Il n’est pas possible d’activer les fonctionnalités vocales dans le cas où l’interface graphique est désactivée).*

**exemple\_questionsxml: *True*** pour afficher et répondre par défaut aux exemples de questions contenues dans le jeu de donnée « *questions.xml* » dès le début du programme sans action de l’utilisateur (lorsque les questions sont toutes traitées, la main est redonnée à l’utilisateur afin qu’il puisse entrer ses propres questions)***,*** configurez ***False*** pour ne pas traiter ce jeu de donnée par défaut.

**exemple\_autres:** Même principe que ci-dessus mais pour des exemples autres que ceux du jeu de donnée « *question.xml* »

# **Détail du code par fonctions**

Le code de ce projet (contenu dans le fichier « projet.py ») est divisé en plusieurs fonctions remplissant différents rôles. Ces fonctions sont réparties en 3 grandes parties. Les fonctions servant au **fonctionnement même du système de questions / réponses**, les fonctions permettant **la gestion de l’interface graphique (GUI)** et enfin les fonctions permettant le **fonctionnement de la reconnaissance et de la lecture vocale**. Nous nous intéresserons ici à la première catégorie, c’est-à-dire toutes les fonctions de base de notre projet implémentant le système de questions / réponses. Nous nous reviendrons plus tard dans ce rapport sur le fonctionnement de l’interface graphique et du système vocal intégré. Les fonctions représentées sur la page suivante sont renseignées dans l’ordre d’exécution, depuis la première fonction appelé dans le « main » de notre programme et prenant en entrée la question de l’utilisateur, à la dernière fonction renvoyant la réponse à la question initialement posée.

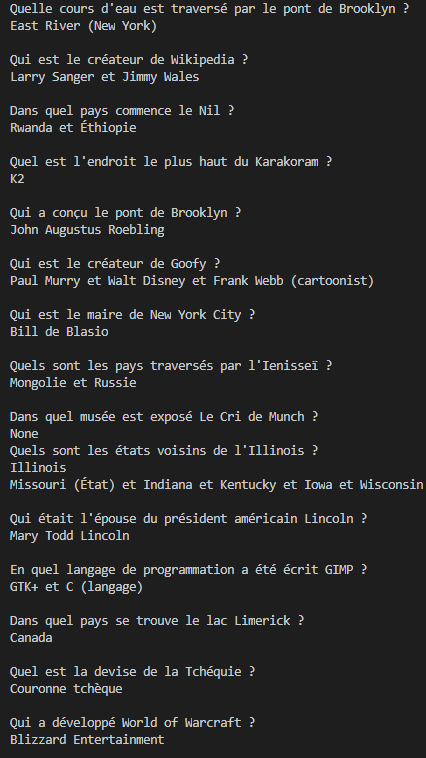
# **Type de questions prises en compte dans notre programme**

**Le tableau suivant donne une liste du type de questions prises en charge par le système que nous avons conçu. La 2ème colonne indique le type de réponse apportée pour chaque question.**

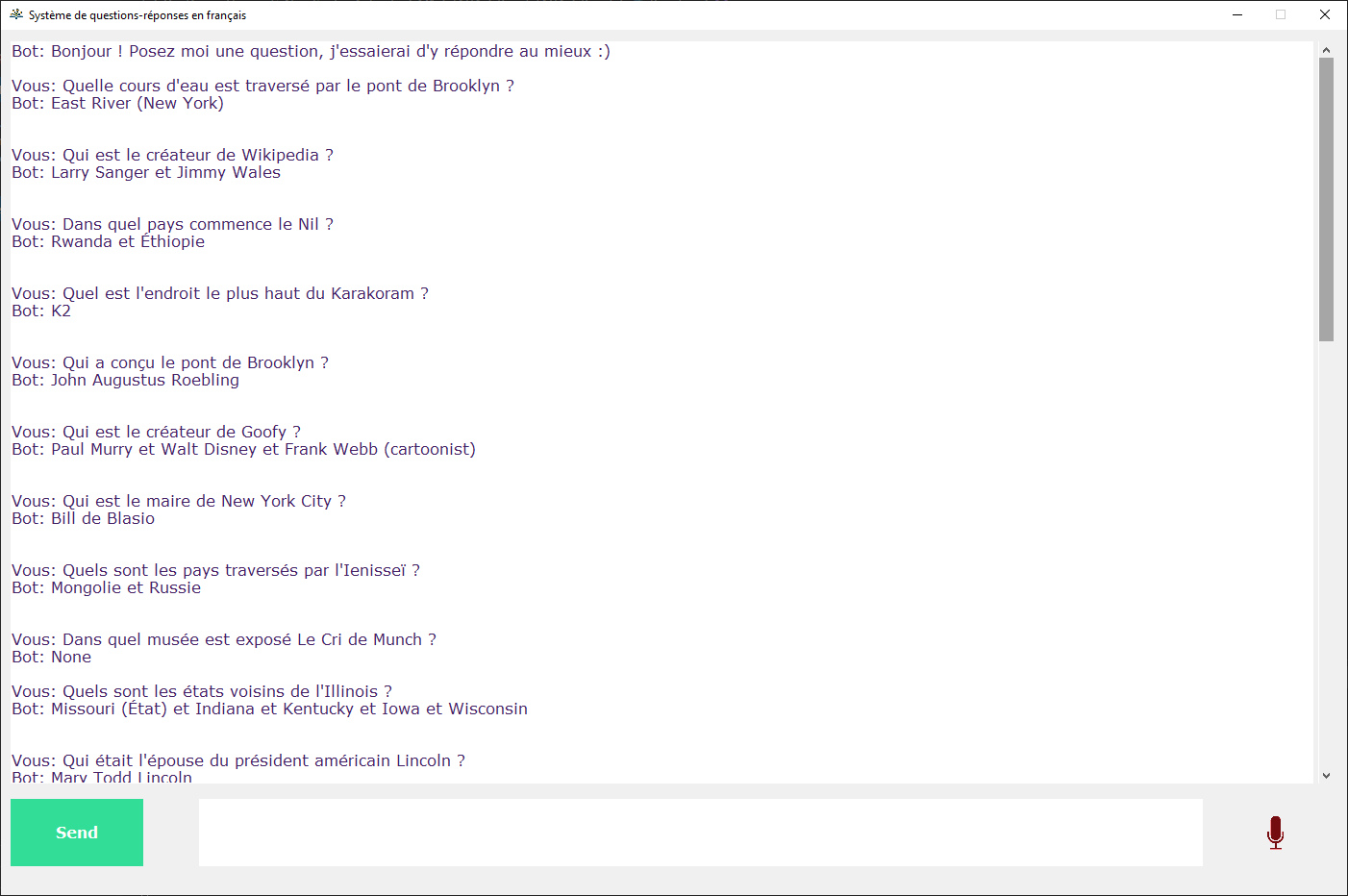
|  |  |
| --- | --- |
| Type de question | Réponse apportée par le système |
| Quelle est la capitale de *nomPays* ? | La capitale de *nomPays* |
| Quel est l’indicatif téléphonique de *nomVille* ? | L’indicatif téléphonique de *nomVille* |
| Dans quel musée est exposé/présenté *nomOeuvreArt* ? | Le nom du musée où *nomOeuvreArt* est exposée |
| Quel est l’endroit le plus haut de *nomLieu* ? | L’endroit le plus élevé de *nomLieu* |
| Qui est/était l’époux/l’épouse/etc. de *nomPersonne* ? | Nom de l’époux/l’épouse/etc. de *nomPersonne* |
| En quel langage de programmation a été écrit *nomLogiciel* ? | Nom du langage de programmation avec lequel *nomLogiciel* a été écrit |
| Quelle est la monnaie de *nomPays* ? | Monnaie de *nomPays* |
| Quand se déroula *nomEvenement* ? | La date de *nomEvenement* |
| Quelle est la date de naissance de *nomPersonne*? | La date de naissance de nomPersonne |
| Quel est le site web de *nomEntreprise* ? | Le site web de nomEntreprise |
| Qui est le maire de *nomVille* ? | Le nom du maire de *nomVille* |
| Qui est le président de *nomPays* ? | Le nom du président de *nomPays* |
| Quels sont les états voisins de *nomEtat* ? | Les états voisins de *nomEtat* |
| Dans quel pays se trouve *nomLieu* ? | Le pays où se trouve *nomLieu* |
| Quels sont les langues officielles de *nomPays* ? | Les langues de *nomPays* |
| A qui appartient *nomEntreprise* ? | Le propriétaire de *nomEntreprise* |
| Quels prix ont été gagné par *nomOrganisation* ? | Les prix gagnés par *nomOrganisation* |
| Qui est le créateur de *nomCreation* ? | Le nom du/des créateur(s) de *nomCreation* |
| Qui a conçu *nomConstruction* ? | Le nom du concepteur de *nomConstruction* |
| Donne-moi tous les acteurs jouant dans le film *nomFilm* | Les noms des acteurs de nomFilm |
| Quel est la cause de décès de *nomPersonne* ? | La cause de décès de *nomPersonne* |
| Qui est *nomPersonne* ? | Description détaillée de *nomPersonne* |
| Combien d’employés a *nomEntreprise* ? | Nombre d’employés embauchés par nomEntreprise |
| Quelle *nomFleuve* est traversée par *nomPont* ? | Le nom du cours d’eau qui traverse *nomPont* |
| Quels sont les pays traversés par *nomFleuve*? | Les noms des pays qui sont traversés par nomFleuve |
| Qui a écrit le livre *nomLivre* ? | Le nom de l’auteur de *nomLivre* |

***Note : les réponses renvoyées correspondent aux données présentes dans la base DBpedia et ne sont pas forcément à jour.***

# **Interface graphique (gui)**



*Version console classique (gui désactivé)*



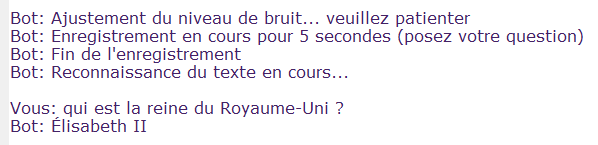
*Version graphique (gui activé)*

En plus du système de questions / réponses dont fait l’objet ce projet, nous avons développé une interface graphique basique pour permettre une utilisation plus conviviale pour l’utilisateur. Nous nous sommes pour cela basés sur la bibliothèque standard de Python **tkinter**. L’interface devrait ainsi s’afficher de la même façon sous une machine pouvant faire tourner un interpréteur Python quel que soit le système d’exploitation. L’interface permet la saisie des questions, la consultation des réponses et l’activation des fonctionnalités vocales (plus de précisions sur ces fonctions dans la section suivante). L’initialisation de l’interface graphique est entièrement effectuée dans la fonction « **demarrage\_gui** » (uniquement si l’option gui est activé, se reporter à la partie « *configuration du projet* » de ce rapport). L’affichage des questions ainsi que des réponses sur l’interface graphique est géré par la fonction « **send** ».

# **Fonctionnalités vocales**

## **Reconnaissance vocale**

La fonctionnalité de reconnaissance vocale que nous avons intégrée permet de comprendre une question posée à l’oral **en français** grâce à un microphone. Pour activer cette fonctionnalité, cliquer sur l’icône rouge représentant un microphone en bas à droite de l’interface graphique. **Merci de noter qu’il est absolument nécessaire d’avoir activer l’interface graphique (*gui*) pour activer l’ensemble des fonctionnalités vocales intégrées.** Afin d’améliorer la reconnaissance de la question, un message indiquera l’ajustement du niveau de bruit avant de vous inviter à poser la question. Un délai de 5 secondes sera alors laissé à l’utilisateur pour poser la question à l’oral. L’ensemble des opérations de reconnaissances vocales sont codées dans la fonction « **voix** ». La conversion de la question orale en français vers du texte s’effectue à l’aide des outils mis à disposition par *Google*.



*Note : Si vous n’apercevez aucune icone de microphone visible sur l’interface gui, merci de vérifier que les fonctionnalités vocales sont activées. Se reporter à la section «****Configuration du projet****» de ce rapport pour plus d’informations.*

## **Lecture vocale**

Lorsqu’une question est posée à l’aide de la reconnaissance vocale, la réponse est apportée **à la fois à l’écrit et à l’oral** via les haut-parleurs de la machine. Les opérations concernant la lecture vocale sont codées dans la fonction « **parler** ». La conversion de la réponse écrite vers une réponse orale en français est également effectuée à l’aide des outils mis à disposition par *Google (Google Text-to-Speech)*. La bibliothèque mise à disposition par *Google* ne permettant que la création d’un fichier *mp3* à partir du texte lu (et non la lecture sur haut-parleur), nous lisons donc ce fichier audio à l’aide d’un lecteur audio externe inclus dans le package **pyglet** avant de directement supprimer le fichier *mp3* du disque.

# **Evaluation de notre système**

Le système conçu répond correctement à 100% des questions proposées dans le jeu de donnée « *question.xml* » (à quelques mises à jour près car les réponses aux questions ont évolué depuis l’écriture du jeu de donnée, et une anomalie dans la base DBpedia renvoyant une date erronée pour la question « Quand se déroula la bataille de Gettysburg ? » [cela fonctionne correctement si remplacement par « bataille de Verdun » par exemple]). Pour cette raison la précision, le rappel et la F-mesure pour chaque question est de 1.

Ce résultat est cependant à relativiser par le fait que nous avons conçu notre système de questions / réponses pour qu’il réponde spécifiquement à toutes les questions de ce jeu de donnée (mais pas exclusivement). En effet si le système devrait très bien s’en sortir sur les questions types mentionnés plus tôt dans ce rapport, il risque d’avoir des réponses beaucoup plus aléatoires ou ne pas donner de réponses du tout sur des questions qui sortent des patterns que nous avons conçus.

# **Améliorations possibles**

Nous avons tenté de livrer le projet le plus abouti possible dans le temps qui nous était imparti mais il est bien sur toujours possible de suggérer des pistes d’améliorations du programme. La principale piste à explorer concernerait les expressions régulières afin d’extraire les bonnes données de la question. En effet, s’il est vrai que des bonnes réponses sont apportées sur la totalité des questions du jeu de données fourni via le fichier « questions.xml », le système conçu n’est pas forcément aussi performant sur des questions traitant complètement d’autres sujets ou même utilisant un vocabulaire différent. Il conviendrait alors d’ajouter de nouvelles expressions régulières afin de matcher de nouveaux types de questions mais aussi de compléter et de généraliser d’avantage celles déjà existantes. Nous n’avons en particulier pas exploité la totalité des entités proposés dans la base *DBpedia* et les nouvelles expressions régulières crées pourraient ainsi exploiter davantage de catégories référencées dans cette base.

Nos expressions régulières se basent d’ailleurs sur une stratégie de type « *exact match* » repérant ainsi des mots précis dans les questions de l’utilisateur, il pourrait être intéressant d’explorer d’autres solutions se basant par exemple la *distance de Levenshtein* ou encore sur la base *WordNet*. Nous avons rapidement expérimenté ce type de stratégie mais nous obtenions de manière générale des réponses plus pertinentes en matchant exactement les mots cherchés. Cela semble particulièrement vrai en français car la plupart des outils repérant les similarités entre les mots semblent, de manière générale, mieux fonctionner en anglais.

Enfin une dernière piste d’amélioration possible consisterait à utiliser une base de données plus à jour pour le renvoi de réponses (les réponses datant, pour la plupart, de quelques années) voir même de développer un outil permettant d’extraire en temps réel les informations disponibles sur le web. L’intégration de « DBpedia Live » au projet pourrait également être intéressante car cet outil permet de relever les modifications qui ont eu lieu sur Wikipédia depuis la dernière mise à jour de *DBpedia*.