Réalisation d'un système Q&A spécialisé dans la qualité des eaux

Maya Touzari

RAPPORT DE STAGE

Réalisation d'un système Q&A spécialisé dans la qualité des eaux

Realisé par :

Maya Touzari

Professeur référent :

Laure Soulier

Encadré par :

Oana Balalau
Ioana Manolescu
Vincent Labbé
Thierry Vilmus

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Le BRGM, Service géologique national, est l'établissement public de référence dans les applications des sciences de la Terre pour gérer les ressources et les risques du sol et du sous-sol [1]. Son action est orientée vers la recherche scientifique, l'appui aux politiques publiques et la coopération internationale. Le BRGM développe notamment depuis plusieurs années une expertise reconnue à l'internationale des systèmes d'information dédiés à l'environnement, en particulier sur le domaine de l'eau, du site web portail de référence aux API web de diffusion des données environnementales, en passant par l'indexation de grands volumes de données et la modélisation prédictive. Parmis les outils developés par le BRGM, le portail Hub'eau qui comporte une dizaine d'API Rest chacune spécialisé dans un domaine en relation avec l'eau (qualité des eaux souterraines, rivières, poisson, etc). Pour permettre aux usagers intéressés par les problématiques environnementales de bénéficier des informations que peuvent offrir ces données, nous proposons de concevoir les premières briques d'un service numérique dont la fonction sera de répondre aux questions des utilisateurs, posées en langage naturel, de façon contextualisée et en se basant sur les données réelles du portail Hub'eau.

Dans ce rapport nous commençons par une brève description de l'état de l'art en matière de chatbots et de systèmes questions-réponses. Par la suite, nous décrirons l'outil Hub'eau développé par le BRGM, nous aborderons les détails d'une des API du portail en précisant la procédure d'interrogation et le format des données. Nous poursuivrons par la description de la démarche entreprise pour le développement du système décrit ci-dessus. Nous terminerons par la présentation des résultats auxquels nous avons pu aboutir.

1. `https://www.brgm.fr/en`

Chapitre 1

État de l'art

1.1 Chatbots et systèmes de dialogue

Les agents conversationnels ou systèmes de dialogue sont des programmes qui ont pour but d'interagir avec un utilisateur en langage naturel, que ce soit par l'écrit ou par la parole. On peut distinguer deux classes d'agents conversationnels. Les agents axés sur les tâches (task-oriented) ayant comme objectif d'aider à accomplir une tâche, par exemple, passer un appel, réserver une place de cinéma, etc. D'autre part, les chatbots sont configurés pour imiter les interactions humain-humain qui ont la particularité de manquer de structure. Les chatbots sont ainsi un moyen de créer des agents orientées tâches plus naturels.

Dans ce chapitre, nous allons introduire les algorithmes fondamentaux des agents conversationnels. Nous commencerons par les chatbots avec les trois principales architectures existantes : à base de règles, systèmes de recherche d'informations, encodeurs-décodeurs. Nous terminerons avec les systèmes de dialogues orientés tâches avec l'architecture frame-based.

1.1.1 Chatbots

1.1.1.1 Architecture basée sur des règles

L'architecture basée sur les règles ou *rule-based* est l'architecture avec laquelle sont construits la majorité des premiers chatbots. La réponse retournée par le système est basée sur un ensemble de règles fixé et prédéfini. Les règles utilisent principalement la reconnaissance de motifs ou de modèles dans le texte en entrée, elles sont codées à la main (*hand-coded*) et ne génèrent donc pas de nouvelles réponses textuelles [12]. Ce type de modèles n'est pas robuste aux fautes d'orthographes ou de grammaire. De plus, une réponse ne se base que sur la dernière question d'une conversation, en supposant qu'elle soit indépendante des questions qui la précèdent. Un chatbot plus naturel permettrait de prendre en considération les parties précédentes de la conversation pour sélectionner une réponse pertinente à l'ensemble du contexte de la conversation [1].

1.1.1.2 Architecture basée sur un corpus

Contrairement aux architecture rule-based, une architecture basée sur un corpus exploite une collection de conversations humaines. Ce type d'architecture est très gourmand en données, nécessitant des millions d'exemples d'interactions humain-humain pour l'apprentissage [17]. Plusieurs ensembles de données conversationnels existent, par exemple, Switchboard qui est un corpus de conversations téléphoniques [4] et le

EMPATHETICDIALOGUES, une collection de plus de 25k conversations en langage naturel [13]. Malgré la taille conséquente de ces collections de données, la plupart des modèles passent par une première étape de pré-entraînement sur des corpus de données textuelles très larges, tirés de Twitter [14], Reddit [15] ou d'autres plates-formes et réseaux sociaux. La plupart de ces systèmes produisent une réponse soit en utilisant des méthodes de récupération (récupérer une réponse à partir d'une collection de réponses appropriée compte tenu du contexte du dialogue) ou des méthodes de génération (en utilisant un modèle de langage ou un encodeur-décodeur pour générer la réponse compte tenu du contexte de dialogue) [5].

Méthodes de récupération

Dans ce type d'architectures, la question de l'utilisateur est vue comme une requête q. Le travail est de retourner la réponse r la plus appropriée, parmi une collection de conversations C. Le choix se fait en donnant un score à chaque réponse potentielle de C et en choisissant celle qui a le score le plus élevé. Les techniques classiques de recherche d'information peuvent être utilisées. Par exemple, l'utilisation d'une représentation tf-idf [1] de q et de C et de la similarité cosine [2] pour le calcul des scores [5].

$$\textit{réponse}(q, C) = \underset{r \in C}{\operatorname{argmax}} \frac{q \cdot r}{|q| \cdot |r|}$$

Une autre méthode très utilisée est de recourir à un réseau de neurones pour construire une représentation (embedding) pour la requête q et les réponses $r \in C$. Le score est, comme pour la méthode précédente qui utilise une représentation tf-idf, calculé grâce à la similarité cosine ou même le produit vectoriel des deux représentations.

$$\textit{réponse}(q, C) = \underset{r \in C}{\operatorname{argmax}} \frac{rep_q \cdot rep_r}{|rep_q| \cdot |rep_r|}$$

$$\textit{réponse}(q, C) = \underset{r \in C}{\operatorname{argmax}}\, rep_q \cdot rep_r$$

où rep_q est la représentation de la requête q et rep_r, pour chaque réponse $r \in C$, est la représentation de la réponse r.

Méthodes de génération

Les systèmes de dialogues basés sur la récupération sont limités à des réponses prédéfinies. Les chatbots qui utilisent des méthodes génératives peuvent, quant à eux, générer du texte à partir des données d'apprentissage.

Le génération de la réponse peut être vue comme une tâche d'encodage-décodage (figure 1.1) où un modèle sequence-to-sequence essaye de traduire la requête utilisateur vers une réponse [18] [21]. Cependant, ce type de modèles a tendance à privilégier les réponses à forte probabilité comme « je sais » ou « OK ». Ce genre de réponses mettent fin à une conversation et la rendent moins naturelle. En conséquence, plusieurs approches ont été utilisées, notamment celles qui tirent partie de l'apprentissage par renforcement pour optimiser les systèmes par les récompenses cumulatives.

1. `https://fr.wikipedia.org/wiki/TF-IDF`
2. `https://fr.wikipedia.org/wiki/Similarit%C3%A9_cosinus`

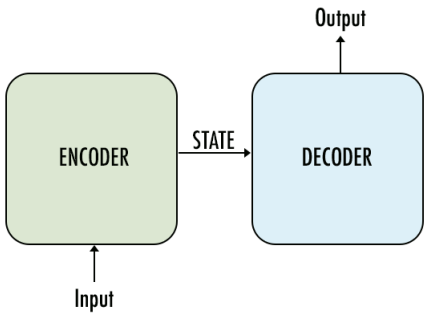

FIGURE 1.1 – Architecture encodeur-decodeur

Méthodes hybrides

Un chatbot axé corpus peut aussi combiner les méthodes basées sur la récupération et les méthodes génératives, par exemple, [19] et [11]. Dans un système basé sur la récupération, les réponses sont plus précises mais directes, d'autre part, les méthodes génératives permettent d'avoir des réponses plus naturelles mais vagues [3]. Dans un modèle hybride, la réponse candidate récupérée par une première étape de recherche d'information et la question utilisateur servent à deux d'entrée au module de génération de réponse (figure 1.2).

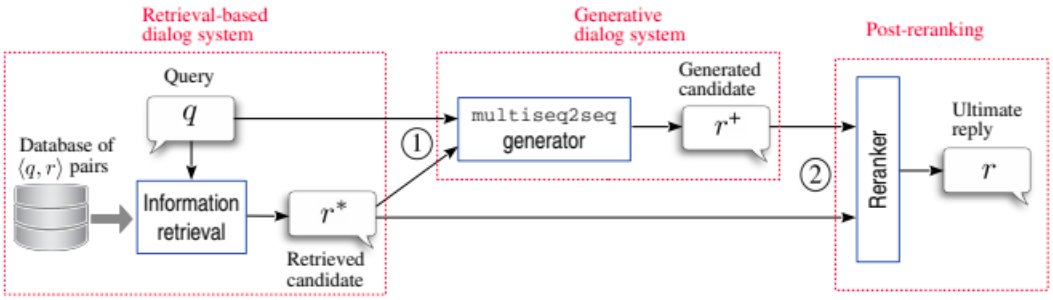

FIGURE 1.2 – Méthode hybride, IR-based et générative (tirée de [19])

1.1.1.3 Architecture hybride

L'architecture d'un chatbot peut aussi être une hybridation de composants basés sur un ensemble de règles et d'autres composants neuronaux (axés corpus). Par exemple, Chirpy Cardinal [9], est un agent de dialogue présenté lors du concours Alexa Prize 2019 [3]. Ce système retourne des réponses produites par une série de générateurs dont une partie sont des modèles de langues neuronaux et l'autre partie des composants basés sur des règles fixes.

3. `https://developer.amazon.com/alexaprize`

1.1.2 Systèmes de dialogue orientés tâches

Les systèmes de dialogue axés sur les tâches sont eux aussi une branche importante des systèmes de dialogue. Contrairement aux chatbots qui se concentrent sur les conversations humaines en langage naturel, un système axé tâches vise à accomplir des tâches que l'utilisateur lui demande de faire. Nous allons présenter l'architecture basée sur les *frames*, nous terminerons avec une architecture plus moderne basée sur l'état du dialogue (*dialogue state*).

1.1.2.1 Architecture basée sur les frames

Tous les systèmes de dialogue axés tâches, que ce soit les architecture simples ou plus modernes, sont basées sur les *frames* (en français cela se traduirait par "cadre" ; nous garderons le terme anglais car plus commun). Un frame est un modèle de connaissance structurée qui représente l'information que le système arrive à extraire de la requête de l'utilisateur. Un ensemble de frames est appelé ontologie de domaine. Un frame est composé d'une collection prédéfinie de slots, chaque slot ayant un ensemble de valeurs possibles et contraintes par une sémantique particulière. Les slots représentent ce que le système de dialogue a besoin de connaître pour comprendre et accomplir une tâche [5]. La table 1.1 montre un exemple de frame et de slots pour un système spécialisé dans la réservation de vols.

Slot	**Type**
ORIGIN CITY	location
DESTINATION CITY	location
DEPARTURE TIME	time
DEPARTURE DATE	date
ARRIVAL TIME	time
ARRIVAL DATE	date

TABLE 1.1 – Exemple de frame, de slots le composant et le type de chaque slot

Structure de contrôle

La structure de contrôle dans un système de dialogue axé frames a comme objectif le remplissage des slots pour ensuite effectuer l'action appropriée pour l'utilisateur (répondre une question, ou réserver un vol). Le remplissage des slots est un autre problème difficile de le domaine du traitement de la langue parlée. Il est généralement défini comme un problème d'étiquetage. L'entrée est la requête de l'utilisateur, constituée d'une séquence de mots, le système assigne pour chaque mot le slot adéquat [3].
Un exemple de structure simple est celle qui pose une question prédéfinie à l'utilisateur pour remplir chaque slot du frame. La table 1.2 reprend l'exemple de la table 1.1 en rajoutant des exemples de modèles de questions qu'une structure peut poser à l'utilisateur pour remplir le slot correspondant.

Slot	Type	Modèle de question
ORIGIN CITY	location	From what city are you leaving ?
DESTINATION CITY	location	Where are you going ?
DEPARTURE TIME	time	When would you like to leave ?
DEPARTURE DATE	date	What day would you like to leave ?
ARRIVAL TIME	time	When do you want to arrive ?
ARRIVAL DATE	date	What day would you like to arrive ?

TABLE 1.2 – Exemple de frame, de slots et modèles de questions

Compréhension du langage naturel

Le composant de compréhension du langage dans un système de dialogue axé frames a comme objectif de définir trois éléments à partir du texte d'entrée. Le premier étant le **domaine** auquel appartient le texte, s'il s'agit d'une demande réservation de vol, d'un appel téléphonique à effectuer, etc. Par la suite, le système a besoin d'identifier l'**intention** de l'utilisateur : veut-il réserver un vol, annuler un vol ou consulter les vols, etc. Enfin, le système termine par le **remplissage** des slots en choisissant le frame correspondant au domaine et à l'intention qu'exprime l'utilisateur à travers sa requête. Le système de dialogue GUS [2] utilise, pour sa part, un ensemble de règles prédéfinies et écrites à la main. Le système Phoenix [22] utilise quant à lui une grammaire sémantique conçue à la main et composée de milliers de règles. Une grammaire sémantique est une grammaire sans contexte dans laquelle le côté gauche de chaque règle correspond aux entités sémantiques exprimées (c'est-à-dire les noms de slot) et le coté droit les modèles de texte correspondant (figure 1.3). De nombreux systèmes de dialogue industriels modernes utilisent l'architecture GUS axée frames, mais utilisent l'apprentissage automatique supervisé pour le remplissage de slots au lieu de règles écrites à la main [5].

SHOW	→	show me | I want | can I see |. . .
HOUR	→	one | two | three | four. . . | twelve (AMPM)
AMPM	→	am | pm
ORIGIN	→	from CITY
DESTINATION	→	to CITY
CITY	→	Boston | San Francisco | Denver | Washington...

TABLE 1.3 – Exemple de grammaire sémantique

Autres composants

Un système de dialogue peut aussi faire appel à un module de **reconnaissance vocale**, qui permet de prendre en entrée des données audio et de les transformer en données textuelles. Le module de **génération de texte** s'occupe de la production du texte qui servira de réponse à l'utilisateur. Les systèmes axés frames ont tendance à générer des réponses basées sur des modèles, complètement ou partiellement prédéfinis.

1.1.2.2 Architecture basée sur l'état du dialogue

Les systèmes de dialogues modernes axés tâches sont basés sur une architecture à base de frames plus sophistiquée appelée architecture à état de dialogue (*dialogue state*). Ces systèmes incluent plusieurs composants, comme illustré à la figure 1.3. Le composant de **compréhension du langage naturel** (*Natural language interpreter*) a pour rôle le remplissage des slots à partir de l'énoncé de l'utilisateur. Le **suivi de l'état du dialogue** (*dialogue state tracker*) maintient l'état actuel du dialogue, incluant les informations sur les questions précédentes de l'utilisateur. La **politique de dialogue** (*dialogue response selection ou dialogue policy*) décide de l'action suivante que doit entreprendre le système en réponse à la demande de l'utilisateur(répondre, suggérer des actions, demander une clarification, etc). Enfin, le composant de **génération de langage naturel** (*Natural language generator*) permet de générer une réponse la plus naturelle possible.

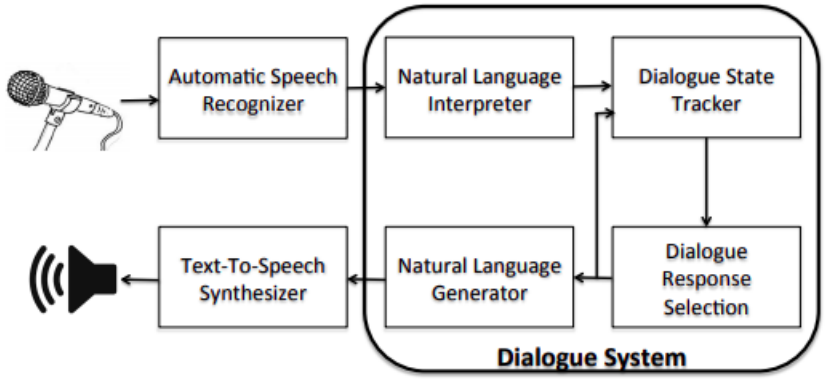

FIGURE 1.3 – Diagramme d'un système basée sur l'état du dialogue (tirée de [16])

1.2 APIs Hub'eau

Hub'eau [4] est un outil de diffusion des données du Service public des Informations sur l'Eau (SIE) [5]. Hub'eau est fondé sur une infrastructure adaptée à la manipulation et au stockage des données massives, et pour mieux répondre aux besoins de ses utilisateurs, Hub'eau organise des compagnes de bêta testing pour recueillir un maximum de retour d'expériences sur les services offerts. Cet outil met à disposition des utilisateurs **dix API** qui facilitent l'accès aux données sous différents formats (JSON, CSV et GeoJSON) : Surveillance des eaux littorales, Hydrobiologie, Prélèvements en eau, Hydrométrie, Température des cours d'eau, Qualité des cours d'eau, Qualité des nappes, Piezométrie, Indicateurs des services, Poissons. Dans la suite, nous allons nous concentrer sur l'API Piézometrie [6]. Nous allons définir des notions de base en relation avec le domaine des eaux souterraines, puis présenter l'API et son utilisation.

4. `https://hubeau.eaufrance.fr/`
5. `https://www.eaufrance.fr/les-donnees-des-sites-eaufrance`
6. `https://hubeau.eaufrance.fr/page/api-piezometrie`

1.2.1 Notions de base

ADES

ADES[7] est la banque nationale d'Accès aux Données sur les Eaux Souterraines qui rassemble sur son site internet public des données quantitatives et qualitatives relatives aux eaux souterraines. Les objectifs de l'ADES sont : de constituer un outil de collecte et de conservation des données sur les eaux souterraines, d'être mobilisable par un large ensemble de partenaires, de permettre les traitements nécessaires à l'action de chacun des partenaires, d'être le guichet d'accès aux informations sur les eaux souterraines, d'avoir un suivi de l'état patrimonial des ressources pour répondre à la politique des eaux souterraines, et enfin d'adopter au niveau national un principe de transparence et d'accessibilité aux données sur les eaux souterraines.

Aquifère

Un aquifère est une formation géologique ou une roche, suffisamment poreuse et/ou fissurée (pour stocker de grandes quantités d'eau) tout en étant suffisamment perméable pour que l'eau puisse y circuler librement. Pour se représenter un aquifère, il faut imaginer un vaste réservoir naturel de stockage d'eau souterraine.

Eaux souterraines

Les eaux souterraines désignent toutes les eaux se trouvant sous la surface du sol en contact direct avec le sol ou le sous-sol et qui transitent plus ou moins rapidement dans les fissures et les pores du sol.

Nappe d'eau souterraine

Une nappe d'eau souterraine est une eau contenue dans les interstices ou les fissures d'une roche du sous-sol qu'on nomme aquifère.

Niveau piézométrique

Le niveau piézométrique caractérise la pression de la nappe en un point donné. Il est exprimé soit par rapport au sol en mètre, soit par rapport à l'altitude zéro du niveau de la mer en mètres **NGF (Nivellement Général Français)**. Autrement dit, c'est le niveau libre de l'eau observé dans un piézomètre.

Piézomètre

Les piézomètres constituent les stations de mesure du niveau piézométrique (niveau d'eau dans la nappe). Un piézomètre est un forage non exploité qui permet la mesure du niveau de l'eau souterraine en un point donné de la nappe. Ce niveau qui varie avec l'exploitation renseigne sur la capacité de production de l'aquifère.

1.2.2 API Piézométrie

Les données de l'API "Piézométrie" sont issues du portail ADES présenté ci-dessus ; elles sont mises à jour quotidiennement. Elles portent sur les mesures de piézométrie (niveau d'eau dans les nappes d'eau

7. `https://ades.eaufrance.fr/`

souterraine), dans toute la France et en provenance de tous les partenaires du système d'information sur l'eau. Les données sont exprimées en **mètres NGF** pour les niveaux, et en **mètres par rapport au repère de mesure** pour les profondeurs. L'API permet de faire trois opérations d'accès aux données :

— L'opération **stations** permet d'accéder aux stations de mesures des niveaux des nappes ou piézomètres.

— L'opération **chroniques** permet d'accéder aux données piézométriques (évolution du niveau des nappes) d'une station de mesure.

— L'opération **chroniques_tr** permet d'interroger en quasi temps-réel plus de 1400 piézomètres qui télé-transmettent leurs données brutes horaires.

L'API peut être interrogée en http ou en https, les données peuvent être transmises sous différents formats : JSON, GeoJSON et CSV (source : hubeau [8]).

Utilisation

Pour pouvoir récupérer les données piézométriques nous utilisons soit le service **chroniques**, soit le service **chroniques_tr**, où la requête doit préciser une station ou des stations de mesure. Les stations sont identifiées par un code unique appelé code BSS. Le service **stations** permet de récupérer tous les codes BSS des stations d'une commune ou d'un département en précisant son code d'identification unique, appelé code INSEE [9]. La chaîne de traitement pour récupérer les données sur les niveaux des nappes situées dans un lieu identifié par son code INSEE est donc la suivante :

1. Interroger le service **stations** de l'API Piézométrie avec le code INSEE du lieu pour récupérer les codes BSS des stations qui sont situées à ce lieu.
2. Interroger le service **chroniques_tr** ou **chroniques** avec les codes BSS des piézomètres pour récupérer les données piézométriques.

Exemple

Nous prenons l'exemple de la commune de Mâcon dont le code INSEE est **71720**. Nous commençons par interroger le service **stations** pour récupérer les codes BSS des piézomètres. Le requête est la suivante :

https ://hubeau.eaufrance.fr/api/v1/niveaux_nappes/stations ?code_commune=71270 &format=json&size=20

La figure 1.4 montre un aperçu du résultat obtenu en format JSON. Ces données montrent qu'il y a 2 piézomètres sur la commune de Mâcon, de codes **06252X0063/PZ1** et **06252X0024/EAU**. À partir des codes BSS, nous pouvons interroger le service **chronique**. Pour récupérer les données du piézomètre de code **06252X0063/PZ1**, la requête utilisée est :

https ://hubeau.eaufrance.fr/api/v1/niveaux_nappes/chroniques ? code_bss=06252X0063/FPZ1&size=5000

La figure 1.5 montre un aperçu du résultat obtenu après l'interrogation du service **chroniques**, en format JSON. Ces données montrent qu'il y a 4272 mesures disponibles sur le piézomètre de code **06252X0063/PZ1**. Les informations les plus importantes récupérées pour chaque mesure sont :

— **code_bss** : code national du piézomètre

— **urn_bss** : lien vers la fiche point d'eau ADES du piézomètre

8. https://hubeau.eaufrance.fr/page/api-piezometrie
9. https://fr.wikipedia.org/wiki/Code_Insee

— **date_mesure** : date et heure de la mesure piézométrique

— **timestamp_mesure** : un nombre entier correspondant au nombre de millisecondes écoulées entre le 1er janvier 1970 à minuit UTC et le moment de la mesure (peut être négatif si la date est antérieure au 01/01/1970)

— **profondeur_nappe** : profondeur du niveau d'eau, en mètres par rapport au repère de mesures

— **niveau_nappe_eau** : côte NGF du niveau d'eau, en mètres (niveau_eau_eau = altitude_repere - profondeur_nappe).


```
{ "count" : 2,
   ...
  "api_version" : "1.1.2",
  "data" : [ {
            "code_bss" : "06252X0063/PZ1",
            "urn_bss" : "...",
            "date_debut_mesure" : "2008-05-21",
            "date_fin_mesure" : "2020-02-17",
            "code_commune_insee" : "71270",
            "nom_commune" : "Mâcon",
            ...,
            "altitude_station" : "214.0",
            "nb_mesures_piezo" : 4258,
            "code_departement" : "71",
            "nom_departement" : "Saône-et-Loire",
            ...
            }, {
            "code_bss" : "06252X0024/EAU",
            "urn_bss" : "...",
            "date_debut_mesure" : "1996-10-31",
            "date_fin_mesure" : "2005-11-14",
            "code_commune_insee" : "71270",
            "nom_commune" : "Mâcon",
            ...,
            "bss_id" : "BSS001PYKW",
            "altitude_station" : "214.0",
            "nb_mesures_piezo" : 1463,
            "code_departement" : "71",
            "nom_departement" : "Saône-et-Loire",
            ...
            } ]
}
```


FIGURE 1.4 – Exemple de réponse du service **stations** en format JSON

```
{
  "count" : 4272,
  "first" : "... ",
  "last" : null,
  "prev" : null,
  "next" : null,
  "api_version" : "1.1.2",
  "data" : [ {
                "code_bss" : "06252X0063/PZ1",
                "urn_bss" : "...",
                "date_mesure" : "2008-05-21",
                "timestamp_mesure" : 1211328000000,
                "niveau_nappe_eau" : 206.89,
                "mode_obtention" : "Valeur mesurée",
                "statut" : "Donnée contrôlée niveau 1",
                "qualification" : "Correcte",
                "code_continuite" : "2",
                "nom_continuite" : "Point lié au point précédent",
                "code_producteur" : "198",
                "nom_producteur" : "Service Géologique Régional Bourgogne (198)",
                "code_nature_mesure" : null,
                "nom_nature_mesure" : null,
                "profondeur_nappe" : 7. 11
              },
              ...,
              {
                "code_bss" : "06252X0063/PZ1",
                "urn_bss" : "...",
                "date_mesure" : "2020-03-02",
                "timestamp_mesure" : 1583107200000,
                "niveau_nappe_eau" : 206.74,
                "mode_obtention" : "Valeur mesurée",
                "statut" : "Donnée contrôlée niveau 1",
                "qualification" : "Correcte",
                "code_continuite" : "2",
                "nom_continuite" : "Point lié au point précédent",
                "code_producteur" : "198",
                "nom_producteur" : "Service Géologique Régional Bourgogne (198)",
                "code_nature_mesure" : null,
                "nom_nature_mesure" : null,
                "profondeur_nappe" : 7. 26
              } ]
}
```


FIGURE 1.5 – Exemple de réponse du service **chroniques** en format JSON

Chapitre 2

Conception

2.1 Architecture générale

En vue de fournir au grand public une information personnalisée et instantanée sur l'environnement, nous proposons de concevoir les premières briques d'un service spécialisé dans l'environnement, dont la fonction sera de répondre aux questions des utilisateurs, posées en langage naturel, de façon contextualisée et en se basant sur les données réelles accessibles via les API web du portail Hub'eau, opéré par le BRGM. Le travail se focalisera sur un périmètre thématique précis : l'eau souterraine. L'atteinte de ce défi passe par la conception d'un système de dialogue basée sur les frames (cf section 1.1.2.1). Pour pouvoir répondre à une question, nous avons besoin d'identifier la requête adéquate vers les API de Hub'eau. La formation de la requête nécessite les trois éléments :

— La localisation, c'est-à-dire, les stations de mesures à interroger.

— La période de temps, c'est-à-dire, le moment auquel ont été prises les mesures.

— La thématique, qui nous permet de choisir quelle API Hub'eau interroger (eau souterraine, rivières. . etc)

Ces trois éléments nous permettent d'identifier trois slots : lieu, temps et thématique. Ces trois éléments permettent ainsi de formuler une requête https et de récupérer les données à partir du portail Hub'eau.

Pour que la réponse soit la plus pertinente possible, il faut effectuer un traitement sur les données récupérées afin de calculer la métrique désirée par l'utilisateur(un maximum, un minimum, une moyenne...). Nous identifions un quatrième slot : métrique. La figure 2.1 illustre de manière générale la chaîne de traitement.

Dans la suite, par faute de temps, nous allons décrire la démarche entreprise pour identifier uniquement les deux slots : lieu (section 2.2) et temps (section 2.3). Nous nous focaliserons sur les eaux souterraines, la thématique est donc fixée à l'API Piézométrie (cf section 1.2.2). À la fin, nous présentons un tableau récapitulatif de toutes les données sans avoir besoin de connaître la métrique désirée. Des travaux futurs pourront aborder les deux slots restant et éventuellement essayer de concevoir un chatbot plus complet ; qui ne se limite pas à une seule requête et arrive à interagir et à prendre en considération tout le contexte d'une conversation en langage naturel.

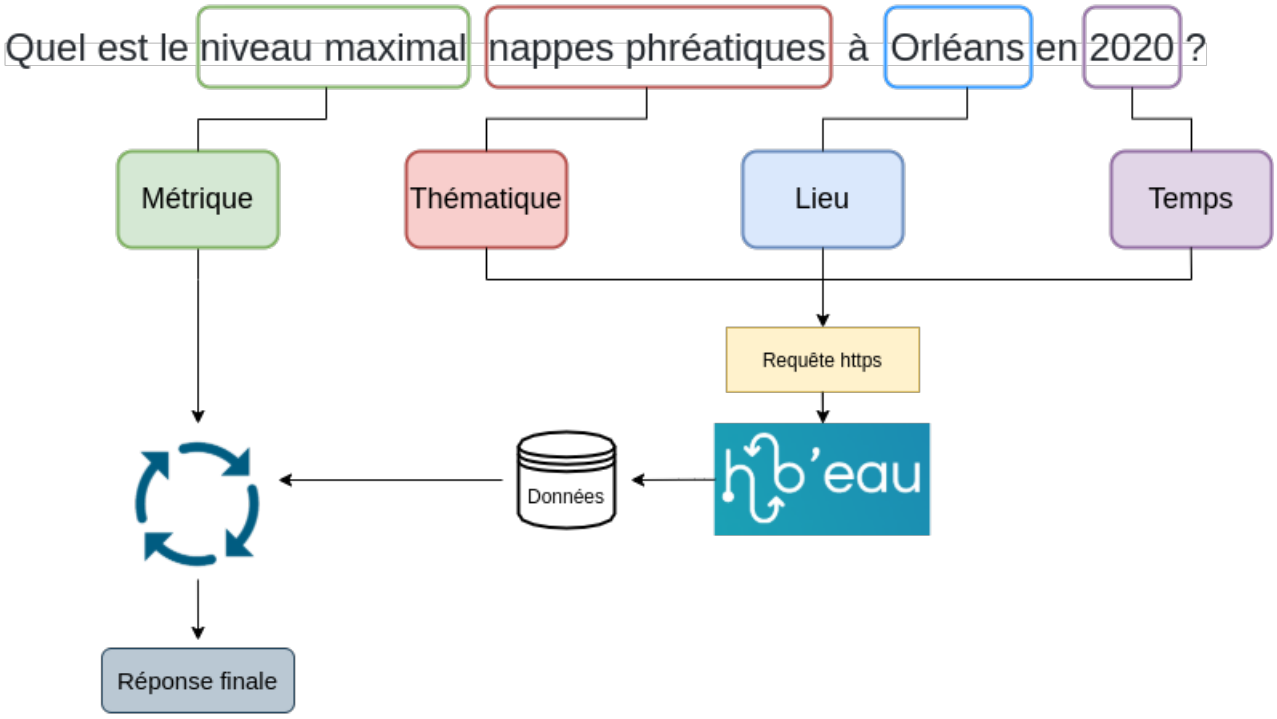

FIGURE 2.1 – Architecture générale

2.2 Période de temps

La première étape de traitements est l'extraction des contraintes de temps que doivent respecter les données piézométriques. La détection des expressions temporelles se fait en utilisant le modèle **HeidelTime** [20] [8]. HeidelTime est un marqueur temporel (temporal tagger) multilingues développé par l'équipe de recherche Database Systems Research Group [1] de l'université de Heidelberg, et ayant reçu des contributions de la part de plusieurs équipes au niveau international [2]. Le modèle extrait les expressions temporelles des documents et les normalise selon la norme d'annotation TIMEX3 [10]. Il peut désormais également être utilisé pour le marquage des temponymes en anglais. Un temponyme étant une phrase avec une portée temporelle bien qu'elle ne soit pas une expression temporelle en soi ou plus formellement, un temponyme est du texte libre qui fait référence à un événement ou un fait ayant une dimension temporelle qui peut uniquement être interprétée compte tenu du contexte et des connaissances de base. Par exemple, *"Finale de la coupe du monde de football 1998"* qui correspond à la date 1998-07-12 [7]. (source : github HeidelTime [3]).

Dans le format TIMEML, chaque expression temporelle est entourée d'une balise TIMEX3 ou TIMEX3-INTERVAL pour les intervalles. Chaque balise contient un attribut **type** qui représente le type de l'expression (DATE, DURATION, SET, etc.) et un attribut **value** qui représente la forme normalisée de l'expression. Par la suite, le traitement du résultat en format TIMEML permet d'avoir un résultat sous forme d'intervalles de temps avec une date de début et une date de fin, en format **yyyy-mm-jj**.

1. `https://dbs.ifi.uni-heidelberg.de/`
2. `https://www.uni-heidelberg.de/en`
3. `https://github.com/HeidelTime/heideltime`

Exemples

Exemple 1 : "Quel est le niveau de la nappe à Orléans le 30 mars 2020 et le 15 avril 2020 ?".
La figure 2.2 montre cette question avec les annotations insérées par HeidelTime (résultat en format TIMEML). La figure 2.3 montre le résultat final après traitement.

```
<?xml version="1. 0"?>
<!DOCTYPE TimeML SYSTEM "TimeML. dtd">
<TimeML>
Quel est le niveau de la nappe à Orléans, Lyon et l'Aube
<TIMEX3 tid="t3" type="DATE"  value="2020-03-30">le 30 mars 2020</TIMEX3> et
<TIMEX3 tid="t4" type="DATE" value="2020-04-15">le 15 avril 2020</TIMEX3>
</TimeML>
```

FIGURE 2.2 – Réponse du modèle HeidelTime pour l'exemple 1 en format TIMEML

```
[   {
    "start_date": "2020-03-30",
    "end_date": "2020-03-31"
    },
    {
    "start_date": "2020-04-15",
    "end_date": "2020-04-16"
    }  ]
```

FIGURE 2.3 – Résultat du traitement pour l'exemple 1

Exemple 2 : Quelle est la qualité de l'eau souterraine dans mon département ?
La question ne contient pas d'expressions temporelles, le résultat du modèle (figure 2.4) ne contient aucune balise TIMEX3, il n y a donc pas de contrainte de temps.

```
<?xml version="1. 0"?>
<!DOCTYPE TimeML SYSTEM "TimeML. dtd">
<TimeML>
Quelle est la qualité de l'eau souterraine dans mon département?
</TimeML>
```

FIGURE 2.4 – Réponse du modèle HeidelTime pour l'exemple 2 en format TIMEML

Exemple 3 : Quelle est la profondeur maximale des nappes à Abancourt en Oise entre le 15 mars et le 30 mars ?
La figure 2.5 montre le résultat obtenu par le modèle en format TIMEML, l'expression temporelle extraite est un intervalle de temps, la balise utilisée est TIMEX3INTERVAL. La figure 2.6 montre le résultat final après traitement.

```
<?xml version="1. 0" ?>
<!DOCTYPE TimeML SYSTEM 'TimeML. dtd'>
<TimeML>
    Quelle est la profondeur maximale des nappes à Abancourt en Oise entre
    <TIMEX3INTERVAL earliestBegin="2022-03-15T00:00:00"
    ↪ earliestEnd="2022-03-30T00:00:00" latestBegin="2022-03-15T23:59:59"
    ↪ latestEnd="2022-03-30T23:59:59">
        <TIMEX3 tid="t1" type="DATE" value="2022-03-15">le 15 mars</TIMEX3>
         et
        <TIMEX3 tid="t2" type="DATE" value="2022-03-30">le 30 mars</TIMEX3>
    </TIMEX3INTERVAL>
    ?
</TimeML>
```

FIGURE 2.5 – Réponse du modèle HeidelTime pour l'exemple 3 en format TIMEML

```
[{
   "start_date": "2021-03-15",
   "end_date": "2021-03-30"
}]
```

FIGURE 2.6 – Résultat du traitement pour l'exemple 3

2.3 Lieux

L'API Hub'eau permet d'accéder aux données piézométriques en utilisant une requête où on précise la station de mesure par son code unique appelé code BSS (cf. chapitre 1.2.2). Dans cette étape, la finalité est d'avoir une liste de code BSS de stations de mesure. Nous allons donc considérer les deux cas :

— L'utilisateur désigne directement un piézomètre spécifique avec son code BSS.

— L'utilisateur désigne seulement un(des) lieu(x), nous considérerons donc toutes les stations situées à ce(s) lieu(x).

2.3.1 À partir de codes BSS

La première étape est de vérifier si la question de l'utilisateur ne contient pas directement des codes BSS de stations. Les codes des stations sont tous de la forme $A_1A_2A_3A_4BCD_1D_2D_3D_4/designation$ où A_i, B et D_i sont des chiffres, C une lettre et $dsignation$ une combinaison de chiffres et de lettres. Par exemple, les codes de stations situées à la commune de Maçon sont $06252X0063/PZ1$ et $06252X0024/EAU$. Nous utilisons une expression régulière pour extraire les codes BSS.

$$\mathbf{[0-9]\{5\}[a-zA-Z][0-9]\{4\}/[a-zA-Z0-9]+}$$

Si la question de l'utilisateur contient bien des codes BSS valides, ils seront directement utilisés pour la requête vers l'API Hub'eau. Sinon, on passe à l'étape suivante où nous essayons de détecter une contrainte de lieu.

2.3.2 À partir de lieux

Si un utilisateur ne précise pas de piézomètre par son code, nous essayons de détécter une autre contrainte de lieu. Un utilisateur peut désigner un lieu en utilisant directement le nom du lieu-dit ou, plus rarement, en utilisant le *gentilé*[4](nom d'habitants d'un certain endroit) correspondant. L'utilisateur peut aussi ne pas préciser du tout de lieu, dans ce cas, nous considérons qu'il fait référence au lieu où il habite (géolocalisation). Nous allons détailler dans la suite la démarche entreprise pour les deux possibilités : noms ou gentilés, et géolocalisation. Enfin, la récupération des stations de mesure respectant les contraintes de lieux détectées est commune aux deux cas.

2.3.2.1 Noms et gentilés

Dans cette étape, nous essayons d'extraire à partir de la requête de l'utilisateur les lieux qui sont des divisions administratives en France, c'est-à-dire, les communes, départements ou régions concernés par sa demande. Cela peut se faire à partir de noms ou de gentilés. La chaîne de traitement peut se résumer en trois étapes :

— Extraire les lieux à partir de noms ou gentilés.

— Classer les lieux en communes, départements ou régions.

— Choisir les lieux pertinents.

2.3.2.1. 1 Étape 1 : Extraction des lieux

Noms :

Pour détecter les **noms** de lieux mentionnés dans la requête, nous faisons recours à la reconnaissance d'entités nommées. Nous utilisons le modèle NER de la librairie de traitement du langage naturel appelée Flair [5]. Il s'agit donc de sélectionner les mots de la requête qui sont tagués par le modèle avec le tag "LOC" et qui représentent des localisations.

Les requêtes utilisateurs ne peuvent concerner que des lieux situés en France, mais il est important de noter que le découpage administratif du territoire français en communes, départements et régions change très régulièrement. On ne peut donc pas envisager une solution où nous construisons une base statique de noms de lieux pour faire une recherche exhaustive de chaque lieu dans le texte de la requête. Néanmoins, du fait que le modèle de reconnaissance nommé ne soit pas tout à fait robuste face à l'absence de majuscule pour la première lettre d'un nom de lieu (sensible à la casse), nous la rajoutons comme étape complémentaire à la reconnaissance d'entités nommées.

Nous avons commencé par recueillir toutes les données récentes sur les noms de communes, départements et régions de France. La source utilisée est le site de l'Insee, l'institut national de la statistique et des études économiques [6]. Par la suite, nous utilisons un modèle d'étiquetage grammatical (Part Of Speech tagging) plus précisément le modèle français de Stanford CoreNLP [7]. Nous sélectionnons les mots étiquetés comme nom (tag "NOUN") et vérifions si ce sont des noms de lieux dans notre base.

Une autre méthode envisageable et qui ne dépende pas d'une base serait de directement interroger l'API gouvernementale de découpage administratif **geo.api.gouv** avec les noms.

4. https://fr.wikipedia.org/wiki/Gentil%C3%A9
5. https://github.com/flairNLP/flair
6. https://www.insee.fr/fr/information/5057840
7. https://stanfordnlp.github.io/CoreNLP/

Gentilés :

Un utilisateur peut désigner un lieu en utilisant le nom des habitants du lieu ou ce qu'on appelle gentilé [8]. Par exemple, *"Quelle est la nappe souterraine* ***parisienne*** *la plus profonde ?"*. Nous avons commencé par créer un dictionnaire lieu-gentilé, qui, pour chaque nom de lieu donne une liste de gentilés utilisés. Pour construire cette base, nous avons utilisé deux méthodes :

— Scrapping du site web habitants.fr [9].

— Scrapping des articles Wikipédia des communes de France. La première étape est de récupérer le code de l'élément Wikidata correspondant à une commune avec une requête SPARQL. Ce code nous permet ensuite de faire le lien avec la page Wikipedia.

Les détails de la construction du dictionnaire se trouvent dans l'annexe 1. En combinant les résultats des deux méthodes, nous avons pu recueillir les gentilés de 106 départements français et de 30422 communes, soit au total 35415 gentilés français. Le dictionnaire n'est pas utilisé en tant que tel, il est transformé de telle sorte à ce que chaque clé soit la forme radicalisée [10] (racine) du gentilé, et que la valeur soit la liste des noms de lieux correspondants (figure 2.7). La radicalisation permet de n'avoir qu'une seule entrée dans le dictionnaire quelque soit le genre (féminin ou masculin) ou le nombre (singulier ou pluriel) du gentilé.

```
{
"communes": {
    "aastois": ["aast"],
    "abainvillois": ["abainville"],
    "abancourtois": ["abancourt"],
    ...
    },
"departements": {
    "aindinois": ["ain"],
    "axonais": ["aisne"],
    "bourbonnais": ["allier"],
    ...
    }
}
```

FIGURE 2.7 – Extrait du dictionnaire de gentilés

En ayant la base de gentilés, le traitement commence par l'application d'un modèle de POS tagging, le modèle français de Stanford CoreNLP[7]. Nous sélectionnons ensuite les mots désignés par le modèle comme adjectifs (tag "ADJ"). Nous vérifions ensuite, pour chacun des adjectifs sélectionnés, s'il s'agit d'un gentilé. Nous utilisons la forme radicalisée pour interroger le dictionnaire transformé. Si l'adjectif est bien un gentilé, tous les lieux correspondant sont sélectionnés.

Divisions administratives :

Grâce aux méthodes précédentes (NER, gentilés) nous obtenons une liste de **noms de lieux**. Ils seront ensuite utilisés pour récupérer les données réelles sur les divisions administratives correspondant, c'est-à-dire, les communes, départements ou régions de France portant ces noms. Nous utilisons L'API de découpage

8. https://fr.wikipedia.org/wiki/Gentil%C3%A9
9. https://www.habitants.fr/index.php
10. https://fr.wikipedia.org/wiki/Racinisation

géographique **geo.api.gouv** [11]. L'API comporte trois services, chacun pour rechercher des informations sur un des trois types de divisions administratives (communes, départements et régions). L'interrogation peut se faire en utilisant soit un nom, soit un code INSEE. Nous allons donc utiliser chacun des noms de la liste, pour interroger chacun des trois services, et récupérer toutes les données sur toutes les communes, départements et régions correspondant, notamment les codes INSEE necessaires pour le service **stations** de l'API Piézométrie de Hub'eau (cf. chapitre 1.2.2).

Exemples :

Exemple 1 : Quel est le niveau de la nappe à l'Aube et à Lyon le 30 mars 2020 ?

La requête comporte deux noms : *"Aube"* et *"Lyon"*. La figure 2.8 montre de résultat de l'extraction. Le resultat en format JSON comporte les champs suivant :

— **locations :** La liste des noms de lieux extraits à partir du texte en entrée.

— **method :** La méthode utilisée pour l'extraction : noms ou noms à partir de gentilés.

— **exact_match :** Pour chaque élément de la liste **locations**, la liste de communes, départements et régions dont le nom correspond exactement à l'élément.

— **similar :** Pour chaque élément de la liste **locations**, la liste des communes, départements et régions dont le nom est composé en partie de l'élément (l'élèment est une sous-chaine du nom).

11. `https://geo.api.gouv.fr/`

```
[{
        "locations": ["aube", "lyon"],
        "method": "<Method. NAME: 2>",
        "exact_match": {
            "aube":{"count": 3,
                    "communes":{"57037": {"codeDepartement": "57",
                                          "codeRegion": "44",
                                          "nom": "Aube"},
                                "61008": {"codeDepartement": "61",
                                          "codeRegion": "28",
                                          "nom": "Aube" }},
                    "departements": {"10": {"codeRegion": "44",
                                            "nom": "Aube"}},
                    "regions": {},
            },
            "lyon":{"count": 1,
                    "communes":{"69123":{"codeDepartement": "69",
                                         "codeRegion": "84",
                                         "nom": "Lyon"}},
                    "departements": {},
                    "regions": {}
            },
        },
        "similar": {
            "aube":{"count": 58,
                    "communes":{"02030":{"codeDepartement": "02",
                                         "codeRegion": "32",
                                         "nom": "Aubencheul-aux-Bois"},
                                ...,
                                "93001":{"codeDepartement": "93",
                                         "codeRegion": "11",
                                         "nom": "Aubervilliers"}},
                    "departements": {},
                    "regions": {}
            },
            "lyon":{"count": 7,
                    "communes":{"03080":{"codeDepartement": "03",
                                         "codeRegion": "84",
                                         "nom": "Cognat-Lyonne"},
                                ...,
                                "85213": {"codeDepartement": "85",
                                          "codeRegion": "52",
                                          "nom": "Rives de l'Yon"}},
                    "departements": {},
                    "regions": {}
            }
        }
    }]
```


FIGURE 2.8 – Résultat de l'extraction de lieux pour l'exemple 1

Exemple 2 : Y'a-t-il de l'eau dans le sous-sol orléanais ?

La liste des gentilés détectés comporte *"orléanais"* qui correspond au gentilés de *"Orléans"*. Il n'y a qu'une seule commune en France portant ce nom. La figure 2.9 montre le résultat du traitement.


```
[{
"locations": {"orléanais": ["orléans"]},
"method": "<Method. DEMONYM: 3>",
"exact_match": {"orléanais":{ "count": 1,
                               "communes": { "45234": { "codeDepartement": "45",
                                                        "codeRegion": "24",
                                                        "nom": "Orléans"}
                                           },
                               "departements": {},
                               "regions": {}
                             }
                },
"similar": { "orléanais": {"communes": {},
                           "departements": {},
                           "regions": {}},
           }
}]
```


FIGURE 2.9 – Résultat de l'extraction de lieux pour l'exemple 2

2.3.2.1. 2 Étape 2 : Classification

L'étape précédente permet d'obtenir, pour chaque expression de lieu (nom ou gentilé), la **liste de communes, départements et régions** correspondant. Parmi les observations faites sur les noms des divisions territoriales en France :

- Il n'existe pas de régions ayant le même nom.
- Il n'existe pas de départements ayant le même nom.
- Il n'existe pas de région ayant le même nom qu'un département.
- Plusieurs communes peuvent avoir le même nom.
- Une commune peut le même nom qu'un département.
- Une commune peut avoir le même nom qu'un département.

Lorsqu'un nom de lieu ne correspond qu'à un seul type de division administrative (commune, département ou région), il n' y a pas d'ambiguïté quant à l'intention de l'utilisateur. Par exemple, le nom *"Abancourt"* ne correspond qu'à des communes.

Par contre, l'ambiguïté existe lorsqu'un même nom correspond à un département et une(des) commune(s) (par exemple, *"Aube"*) ou à une région et une(des) commune(s) (par exemple, *"Bretagne"*) . Il devient nécessaire de prendre une décision sur la granularité à considérer et qui corresponde à l'intention exprimée par la requête de l'utilisateur. Une manière d'y remédier serait de demander directement à l'interlocuteur de clarifier sa requête. Néanmoins, nous choisirons de nous baser uniquement sur le texte en entrée sans avoir recours aux questions de clarification. La suite décrit la démarche et l'ensemble de règles utilisées pour classifier chaque lieu en l'un des trois types de divisions administratives. Nous détaillerons plus particulièrement le cas où un nom peut représenter un département et une (des) commune(s) . Les règles sont similaires lorsqu'il s'agit de

classifier un gentilé ou lorsque nous nous trouvons dans le cas où le nom correspond à une région et une(des) commune(s).

La classification du nom de lieu se base sur le contexte de la requête de l'utilisateur. Nous utilisons aussi la notion d'analyse des dépendances (*dependency parsing*) [6] . Cette dernière nous aide à construire un arbre à partir du texte, avec des tags utilisés pour déterminer la relation entre les mots. L'algorithme général pour classer un lieu qui peut représenter une commune ou un département est le suivant :

1. En premier lieu, nous utilisons une expression régulière pour déterminer si la requête contient le mot "département".

2. Si la requête contient le mot "département" :

2. 1. Nous utilisons le modèle de dependency parsing de Stanford CoreNLP [12] pour détecter tous les mots de la requête qui sont sémantiquement dépendants du mot "département".

2. 2. Si le nom du lieu fait partie des dépendances du mot "département", le lieu est classé comme département.

2. 3. Sinon, passer à l'étape 3.

3. Sinon, l'ambiguïté ne s'est pas encore éclaircie, nous ne pouvons pas trancher si le lieu est une commune ou un département dans le contexte de la requête. Dans cette étape, nous essayons de tirer partie des autres lieux dans la requête qui ont pu être classifiés afin de déduire le type du lieu courant. Nous avons deux cas possibles :

3. 1. L'ensemble des communes et le départements correspondant au nom sont situés dans des régions différentes :

Si en citant le nom du lieu dans sa question, l'utilisateur visait une commune, le fait de préciser la région où se situe cette commune permettrait facilement de deviner qu'il s'agit bien de la commune et non du département, puisque ce dernier est forcément situé dans une autre région. De même s'il visait le département et qu'il précise sa région, il serait facile de deviner qu'il parle du département et non de la commune. Cette observation n'est, bien sûr, possible que lorsque l'ensemble des communes et le département sont dans des régions différentes. Nous utilisons donc la liste de lieux qui ont pu être classifiés avec les étapes précédentes pour classer les lieux où l'ambiguïté persiste. Par exemple, le nom *"Vaucluse"* correspond à une commune au département du *"Doubs"* en région de *"Bourgogne-Franche-Comté"* et il correspond aussi à un département en région *"Provence-Alpes-Côte d'Azur"*. Dans une requête de la forme *"... à Vaucluse en Bourgogne-Franche-Comté ... "* ou de la forme *"... Vaucluse au département du Doubs ... "*, il est facile de déduire que l'utilisateur parle de la commune de *"Vaucluse"* et non du département, grâce aux autres lieux mentionnés dans le texte de la requête.

3. 2. Si une des communes et le département correspondant à ce lieu sont situés dans la même région, connaître cette dernière n'apporte pas plus d'éclaircissement. Si parmi les autres lieux classés (départements ou régions) aucun ne correspond à une division englobant une des communes ou le département, cela n'apporte pas plus d'éclaircissement non plus. Dans ces deux derniers cas, le lieu est par défaut classé comme commune.

Comme précisé auparavant, l'algorithme précédent répond au cas où un nom de lieu correspond à une(des) communes et un département, les règles associées aux autres cas étant similaires. La sortie de l'algorithme global donne **trois listes : une liste de communes, de départements et de régions**.

Exemples :

Cette partie reprend les exemples de la partie extraction (cf. Section 2.3.2.1. 1)

12. `https://stanfordnlp.github.io/CoreNLP/`

Exemple 1 : Quel est le niveau de la nappe à l'Aube et à Lyon le 30 mars 2020 ?

La figure 2.8 montre de résultat de la première étape d'extraction. Le choix pour *"Lyon"* est trivial ; il n y a qu'une commune portant ce nom donc correspond forcément à une commune. Le choix pour *"Aube"* est ambigu ; il peut correspondre à une commune ou à un département. Dans cette exemple, la question ne comprend pas de clarifications sur quelle division considérer, *"Aube"* est alors classé comme commune. La figure 2.10 montre le résultat de classification. *"Aube"* est bel et bien classé comme commune car le résultat ne comprend que les communes portant ce nom sous le champ **communes**.


```
{
  "Communes": {
    "aube": {
        "57037": {"codeDepartement": "57", "codeRegion": "44", "nom": "Aube"},
        "61008": {"codeDepartement": "61", "codeRegion": "28", "nom": "Aube"},
    },
    "lyon": {
        "69123": {"codeDepartement": "69", "codeRegion": "84", "nom": "Lyon"},
    },
  },
  "Departements":{},
  "Regions":{}
}
```


FIGURE 2.10 – Résultat de la classification pour l'exemple 1

Exemple 2 : Quel est le niveau de la nappe à au département de l'Aube et à Lyon le 30 mars 2020 ?

Le résultat de la première étape d'extraction est similaire à l'exemple 1 (figure 2.8). Le choix pour *"Lyon"* reste trivial. Dans cette exemple, l'utilisateur précise qu'il parle du département *"Aube"* et non de la commune. La figure 2.11 montre le résultat de la classification. *"Lyon"* est classé comme commune et *"Aube"* classé comme département.


```
{
  "Communes": {
        "lyon": {
           "69123": {"codeDepartement": "69", "codeRegion": "84", "nom": "Lyon"}},
  },
  "Departements":{
        "aube": {
           "10": {"codeRegion": "44", "nom": "Aube"}},
  },
  "Regions":{}
}
```


FIGURE 2.11 – Résultat de la classification pour l'exemple 2

Exemple 2 : Y'a-t-il de l'eau dans le sous-sol orléanais ?

La figure 2.9 montre le résultat de la première étape d'extraction. La figure 2.12 montre le résultat de la classification, le choix est trivial pour *"Orléans"* car il n y a qu'une seule commune portant ce nom.

```
{
  "Communes": {
        "orléanais": {"45234": {"codeDepartement": "45",
                                "codeRegion": "24",
                                "nom": "Orléans"}},
  },
  "Departements":{},
  "Regions":{}
}
```

FIGURE 2.12 – Résultat de l'extraction de lieux pour l'exemple 2

2.3.2.1. 3 Étape 3 : Choix de pertinence

Comme précisé précédemment, la classification permet de classer chaque nom ou gentilé en l'une des trois divisions : commune, département ou région. Tous les lieux qu'un utilisateur mentionne dans sa question ne doivent pas forcément être utilisés pour sélectionner les stations de mesures. Par exemple, *"J'habite à Loury dans le Loiret, à quelle profondeur faut-il que je creuse un puits dans mon jardin pour avoir de l'eau ?"*, la question comporte deux lieux : la commune du *"Loury"* et le département du *"Loiret"* mais le deuxième ne sert que de précision sur le premier. Il est plus correct de considérer uniquement la commune du *"Loury"* comme contrainte de lieu. Cette observation ne se fait que lorsqu'après classification, nous avons des lieux qui appartiennent à des types différents. Ainsi, il est possible qu'un lieu ne soit mentionné que pour préciser la localisation d'un autre de granularité inférieure. D'autre part, dans le cas des communes, un même nom peut correspondre à plusieurs communes, là aussi, le fait de préciser le département ou la région permet d'en sélectionner une. Dans cette partie, nous essayons d'éliminer tous les lieux qui ne servent que de complément et éventuellement nous en servir pour sélectionner les communes dans le deuxième cas mentionné auparavant.

2.3.2.2 Géolocalisation

Si les deux méthodes citées précédemment ne donnent pas de résultats, nous considérons que l'utilisateur ne spécifie pas de lieu, nous prenons donc le lieu où il se situe géographiquement. Pour localiser un utilisateur, nous utilisons son adresse IP et l'API **ipinfo** [13]. Nous obtenons les coordonnées (longitude et latitude) de l'utilisateur. La figure 2.13 montre le résultat de la géolocalisation.

Comme pour le cas des noms et gentilés, le problème de la granularité de la division administrative à considérer (commune, département ou région) se pose même pour la géolocalisation. Pour décider, nous utilisons une expression régulière pour extraire l'un des deux mots "département" ou "région". Si le résultat est négatif, nous prenons la granularité la plus fine par défaut (commune). Après prise de décision, l'interrogation de l'API **geo-api. gouv** [14] avec la longitude et la latitude permet d'obtenir le lieu correspondant (commune, département ou région).

13. https://ipinfo.io/
14. https://geo.api.gouv.fr/

```
[{
    "locations":{ "location": "Paris",
                  "lat": 48. 8566969,
                  "long": 2. 3514616  },
    "method":"<Method. GEOLOCATION: 1>"
}]
```


FIGURE 2.13 – Résultat de la géolocalisation

Exemple

Quelle est la profondeur maximale des eaux souterraines chez moi ?

La géolocation donne les coordonnées de la figure 2.13, la figure2.14 montre le résultat de la classification. La requête ne précise pas la division à considérer ; par défaut, nous considérons la commune correspondant aux coordonnées.


```
{
  "Communes": {"75056": {"codeDepartement": "75",
                         "codeRegion": "11",
                         "nom": "Paris"}
               },
  "Departements": {},
  "Regions":{}
}
```


FIGURE 2.14 – Résultat de l'exemple

2.3.2.3 Récupération des stations de mesures

L'extraction des contraintes de lieux, que ce soit avec géolocalisation, noms ou gentilés nous permet d'avoir une liste de divisons administrative identifiées par leur code INSEE [15]. Les codes permettent d'interroger le le service **stations** de l'API piézometrie de Hub'eau et de récuperer tous les codes BSS des piézomètres respectant les contraintes de localisation (cf API Piézométrie 1.2.2).

2.4 Mesures piézométriques

À la fin des deux étapes précédentes, nous aurons une liste de codes BSS de piézomètres ainsi qu'une liste d'intervalles de temps. Grâce à ces deux éléments, nous pouvons interroger les service **chroniques** de l'API Piézométrie. Le but étant de récupérer toutes les mesures prises sur les piézomètres indiqués et durant les périodes de temps spécifiées (cf API Piézométrie 1.2.2). Grâce aux mesures, nous construisons un tableau récapitulatif des données, pour chaque période de temps et pour chaque commune, département ou région des contraintes de lieu. La table 2.1 montre un exemple de tableau récapitulatif des mesures pour la commune d'Orléans pendant toute l'année 2020.

15. https://fr.wikipedia.org/wiki/Code_Insee

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)				Prof. nappe (/au sol)			
					Min	Max	Avg	last	Min	Max	Avg	last
Orléans (45234)	03635X0545/PZ1	365	2020-01-01	2020-12-30	88. 08	90. 02	88. 57	89.15	3. 32	5. 26	4. 76	4.19
	03636X1060/PZ2	0	-	-	-	-	-	-	-	-	-	-
	03636X1061/PZ3	362	2020-01-01	2020-12-30	89. 33	91. 14	89. 09	90.34	3. 38	5. 19	4. 65	4.18
	03636X1062/PZ4	365	2020-01-01	2020-12-30	89. 49	91. 07	89. 93	90.43	3. 89	5. 47	5. 02	4.53
	03982X1045/F	365	2020-01-01	2020-12-30	90. 23	91. 39	90. 62	91.0	2. 86	4. 02	3. 62	3.25

TABLE 2.1 – Exemple de tableau récapitulatif des mesures piézométriques.

2.5 Résultats

Cette section présente des exemples de questions avec les résultats finaux offerts par le système. Nous considérons que l'utilisateur habite à Paris et que la date du jour est le 2021-08-28.

Exemple 1 : Quel était le niveau de la nappe phréatique à la station piézométrique 03635X0545/PZ1 le 12 janvier 2019 ?

Contraintes de temps :

Date de debut	Date de fin
2019-01-12	2019-01-13

Codes BSS mentionnés :
03635X0545/PZ1

Tableaux récapitulatifs :
Tableau recapitulatif de la periode du 2019-01-12 au 2019-01-13 :

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Orléans (45234)	03635X0545/PZ1	1	2019-01-12	2019-01-12	88.39	88.39	88.39	4.95	4.95	4.95

Exemple 2 : Quel est le niveau actuel de la nappe phréatique au Centre-val de Loire ?

Contraintes de temps :

Date de debut	Date de fin
2021-08-28	2019-08-29

Divisions administratives selectionnées :

Nom	Type de division	Code INSEE
Loiré	commune	49178
Centre-Val de Loire	region	24

Liste des piézomètres :

Division	Piézomètres
Loiré(49178)	Aucun piézomètre
Centre-Val de Loire (24)	03296X1032/PC1, 04002X0015/F1, 03646X0086/F1, 05716X0008/F1, 03961X0070/P1, 05471X0070/P1, 03975X0070/P1, 03616X0070/P1, 04307X0001/P1, 03981X0101/P1, 03983X0101/P1, 05445X0081/P1, 02913X0002/P1, 03284X0012/P1, 05194X0012/P1, 03955X0032/P1,...

Tableaux récapitulatifs :
Tableau recapitulatif de la periode du 2021-08-28 au 2019-08-29

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Loiré (49178)	-	-	-	-	-	-	-	-	-	-
Centre-Val de Loire(24)	03296X1032/PC1	0	-	-	-	-	-	-	-	-
	04002X0015/F1	0	-	-	-	-	-	-	-	-
	03646X0086/F1	0	-	-	-	-	-	-	-	-
	...									

Exemple 3 : **J'habite à Loury dans le Loiret, à quelle profondeur faut-il que je creuse un puits dans mon jardin pour avoir de l'eau ?**

Aucune expression de temps detectée.

Divisions administratives selectionnées :

Nom	Type de division	Code INSEE
Loury	commune	45188
Jardin	commune	38199

Liste des piézomètres :

Division	Piézomètres
Loury(45188)	03634X0049/P1
Jardin(38199)	Aucun piézomètre

Tableaux récapitulatifs :
Tableau récapitulatif de toutes les mesures prises :

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Loury (45188)	03634X0049/P1	1730	1974-01-01	1978-12-01	107.16	109.08	107.8	33.37	35.29	34.64
Jardin (38199)	-	-	-	-	-	-	-	-	-	-

Exemple 4 : **Depuis 2015, quelle est l'évolution du niveau moyen de la nappe au point 01937X0054/F ?**

Contraintes de temps :

Date de debut	Date de fin
2015-01-01	2021-08-28

Codes BSS mentionnés :
01937X0054/F

Tableaux récapitulatifs :
Tableau recapitulatif de la periode du 2015-01-01 au 2021-08-28 :

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Villers-en-Haye(54573)	01937X0054/F	2427	2015-01-01	2021-08-23	190.69	203.33	196.454	0.13	12.77	7.005

Exemple 5 : **Dans un rayon de 10 km autour d'Abancourt en Oise, à quel endroit la nappe est-elle la plus proche du sol les 10 dernières années ?**

Contraintes de temps :

Date de debut	Date de fin
2011-08-28	2021-08-28

Divisions administratives selectionnées :

Nom	Type de division	Code INSEE
Abancourt	commune	60001

H **Liste des piézomètres :**

Division	Piézomètres
Abancourt(60001)	00608X0039/P

Tableaux récapitulatifs :

Tableau recapitulatif de la periode du 2011-08-28 au 2021-08-28

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Abancourt (60001)	00608X0039/P	-	-	-	-	-	-	-	-	-

Exemple 6 : **En region francilienne, où et quand a-t-on atteint le niveau de nappe le plus bas ?**

Aucune expression de temps detectée.
Divisions administratives selectionnées :

Nom	Type de division	Code INSEE
Île-de-France	region	11

Liste des piézomètres :

Division	Piézomètres
Île-de-France (11)	02201X0105/H2(S7), 1837A0569/SEINE-, 01836A0284/S2/500, 01833C0243/10, 01833C0349/0110, 01837A1190/SC110, 01837A0158/F110, 01837A0750/PZ110, 01833D0553/SA10, 01837A0684/SC10, 01833D0344/SD10, 01836B0260/SCE10, 01837B1024/F10, 01837A1961/851F10,...

Tableaux récapitulatifs :
Tableau recapitulatif de la periode du 2021-08-23 au 2021-08-30

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Île-de-France (11)	02201X0105/H2(S7)	1	1971-09-22	1971-09-22	69.05	69.05	69.05	2.7	2.7	2.7
	01837A0569/SEINE-	1	1972-06-02	1972-06-02	15.05	15.05	15.05	12.05	12.05	12.05
	02946X0012/F200	1	1960-11-23	1960-11-23	57.66	7.66	57.66	1.2	1.2	1.2
	...									

Exemple 7 : Quelle a été la qualité de l'eau souterraine dans mon département ce mois ?

Contraintes de temps :

Date de debut	Date de fin
2021-08-01	2021-08-31

Divisions administratives selectionnées : Utilisateur geolocalisé.

Nom	Type de division	Code INSEE
Paris	departement	75

Liste des piézomètres :

Division	Piézomètres
Paris (75)	01837A0569/SEINE-, 01836A0284/S2/500, 01833C0243/10, 01837A0404/10, 01833C0349/0110, 01837A1190/SC110, 01837A0158/F110, 01837A0750/PZ110, 01833D0553/SA10, 01837A0684/SC10, 01833D0344/SD10, 01836B0260/SCE10, 01837B1024/F10, 01837A1961/851F10,...

Tableaux récapitulatifs :
Tableau recapitulatif de toutes les mesures prises :

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Paris (75)	01837A0569/SEINE-	-	-	-	-	-	-	-	-	-
	01836A0284/S2/500	-	-	-	-	-	-	-	-	-
	...									

Exemple 8 : Quel a été le minimum atteint par le niveau de la nappe phréatique sur le département de l'Aube pendant chaque mois de 2019 ?

Contraintes de temps :

Date de debut	Date de fin
2019-01-31	2019-01-01
2019-02-01	2019-02-28
2019-02-01	2019-02-28
2019-03-01	2019-03-31
2019-04-01	2019-04-30
2019-05-01	2019-05-31
2019-06-01	2019-06-30
2019-07-01	2019-07-31
2019-08-01	2019-08-31
2019-09-01	2019-09-30
2019-10-01	2019-10-31
2019-11-01	2019-11-30
2019-12-01	2019-12-31

Divisions administratives selectionnées :

Nom	Type de division	Code INSEE
Aube	departement	10

Liste des piézomètres :

Division	Piézomètres
Aube (10)	03001X0160/T200, 03001X0183/T300, 02994X0141/T110, 02994X0209/T210, 03001X0188/T310, 02994X0325/D410, 03001X0073/D10, 02623X0022/S10, 03001X0137/T120, 02994X0219/T220, 02994X0292/T320, 03001X0205/D420, 02994X0099/D20, 02621X0019/S20,...

Tableaux récapitulatifs :

Tableau recapitulatif de la periode du 2019-01-01 au 2019-01-31

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	30	2019-01-01	2019-01-30	125.4	128.52	127.37	17.68	20.8	18.82
	03336X0001/S1	30	2019-01-01	2019-01-30	137.04	137.17	137.09	1.43	1.56	1.50
	...									

Tableau recapitulatif de la periode du 2019-02-01 au 2019-02-28

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	27	2019-02-01	2019-02-27	128.71	130.11	129.68	16.09	17.49	16.51
	03336X0001/S1	27	2019-02-01	2019-02-27	136.99	137.16	137.06	1.44	1.61	1.53
	...									

Tableau recapitulatif de la periode du 2019-03-01 au 2019-03-31

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	30	2019-03-01	2019-03-30	129.93	130.13	129.99	16.07	16.27	16.20
	03336X0001/S1	30	2019-03-02	2019-03-31	136.97	137.18	137.06	1.42	1.63	1.53
	...									

Tableau recapitulatif de la periode du 2019-04-01 au 2019-01-30

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	30	2019-04-01	2019-04-30	128.94	129.94	129.50	16.26	17.26	16.69
	03336X0001/S1	30	2019-04-01	2019-04-30	136.99	137.11	137.04	1.49	1.61	1.55
	...									

Tableau recapitulatif de la periode du 2019-05-01 au 2019-05-31

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	31	2019-05-01	2019-05-31	128.6	129.02	128.82	17.18	17.6	17.37
	03336X0001/S1	30	2019-05-01	2019-05-30	137.02	137.24	132.65	1.36	1.58	1.47
	...									

Tableau recapitulatif de la periode du 2019-06-01 au 2019-06-30

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	30	2019-06-01	2019-06-30	127.29	128.64	128.017	17.56	18.91	18.183
	03336X0001/S1	29	2019-06-01	2019-06-29	136.96	137.11	132.46	1.49	1.64	1.51
	...									

Tableau recapitulatif de la periode du 2019-07-01 au 2019-07-31

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	31	2019-07-01	2019-07-31	124.6	127.2	125.82	19.0	21.6	20.37
	03336X0001/S1	30	2019-07-01	2019-07-30	136.73	136.94	132.40	1.66	1.87	1.72
	...									

Tableau recapitulatif de la periode du 2021-08-01 au 2021-08-31

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	30	2019-08-01	2019-08-30	123.56	124.53	123.94	21.67	22.64	22.25
	03336X0001/S1	30	2019-08-01	2019-08-30	136.68	136.84	136.77	1.76	1.92	1.82
	...									

Tableau recapitulatif de la periode du 2019-09-01 au 2019-09-30

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	30	2019-09-01	2019-09-30	122.94	123.5	123.19	22.7	23.26	23.0
	03336X0001/S1	30	2019-09-01	2019-09-30	136.5	136.72	136.59	1.88	2.1	2.0
	...									

Tableau recapitulatif de la periode du 2019-10-01 au 2019-01-31

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	30	2019-10-01	2019-10-30	122.93	123.39	123.06	22.81	23.27	23.13
	03336X0001/S1	30	2019-10-01	2019-10-30	136.62	137.2	137.05	1.4	1.98	1.54
	...									

Tableau recapitulatif de la periode du 2019-11-01 au 2019-11-30

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	29	2019-11-01	2019-11-29	123.5	126.82	124.93	19.38	22.7	21.26
	03336X0001/S1	29	2019-11-02	2019-11-30	137.05	137.24	137.11	1.36	1.55	1.48
	...									

Tableau recapitulatif de la periode du 2019-12-01 au 2019-12-31

Lieu	Code station	Nombre de mesures	Date plus ancienne	Date plus récentes	Niv. enregistré (altitude/mer)			Prof. nappe (/au sol)		
					Min	Avg	Max	Min	Avg	Max
Aube (10)	03001X0160/T200	0	-	-	-	-	-	-	-	-
	...									
	02615X0020/S1	30	2019-12-01	2019-12-30	127.11	132.17	129.06	14.03	19.09	17.13
	03336X0001/S1	30	2019-12-01	2019-12-30	137.05	137.24	137.12	137.07	1.36	1.55
	...									

Chapitre 3

Conclusion et perspectives

Un accès plus naturel et simplifié aux données environnementales offertes par différents organismes gouvernementaux est un objectif important à atteindre, surtout en des temps où les problèmes environnementaux dus au réchauffement climatique sont devenus conséquents. Grâce au travail réalisé durant ce stage, nous avons pu mettre en place les premiers éléments d'un système dont le but finale sera d'atteindre en partie cet objectif ; de pouvoir répondre de manière naturelle aux questions liées à la qualités des eaux environnantes. Le travail nous a aussi permis d'avoir plusieurs perspectives quant aux améliorations et au futurs travaux pour le compléter. Notamment, le remplissage des deux slots qui n'ont pas été traités : la métrique et le thème. Aussi, la construction d'un dataset de requêtes annotées qui permettrait d'envisager la conception de modèles neuronaux plus sophistiqués et plus important encore, de calculer une métrique qui permettrait d'évaluer le système résultant.

Annexes

.1 Annexe 1 : Construction du dictionnaire de gentilés

Comme mentionné précédemment,une manière de faire référence à un lieu est de mentionner son gentilé (nom des habitants). Dans cette appendice nous allons décrire comment nous avons construit un dictionnaire gentilé-lieu où chaque clé est un gentilé et la valeur associée le nom du lieu correspondant, et ce à partir de différentes ressources sur le web. Nous avons utilisé principalement deux ressources : le site web habitants.fr et Wikipedia. La suite sera donc divisée en deux partie, chacune expliquant la démarche entreprise pour collecter les données à partir des deux ressources.

.1.1 À partir de habitants.fr

Le site web habitants.fr [1] a été créé en 2004, c'est un site spécialisé dans le recensement des gentilés français. Le site comporte, pour chaque commune et département en France, une page dédiée contenant la liste de gentilés correspondant. La figure 1 montre un exemple de page extraite de ce site web. Le traitement consiste à l'application d'expressions régulière sur le code HTML de la page pour extraire les gentilés mentionnés.

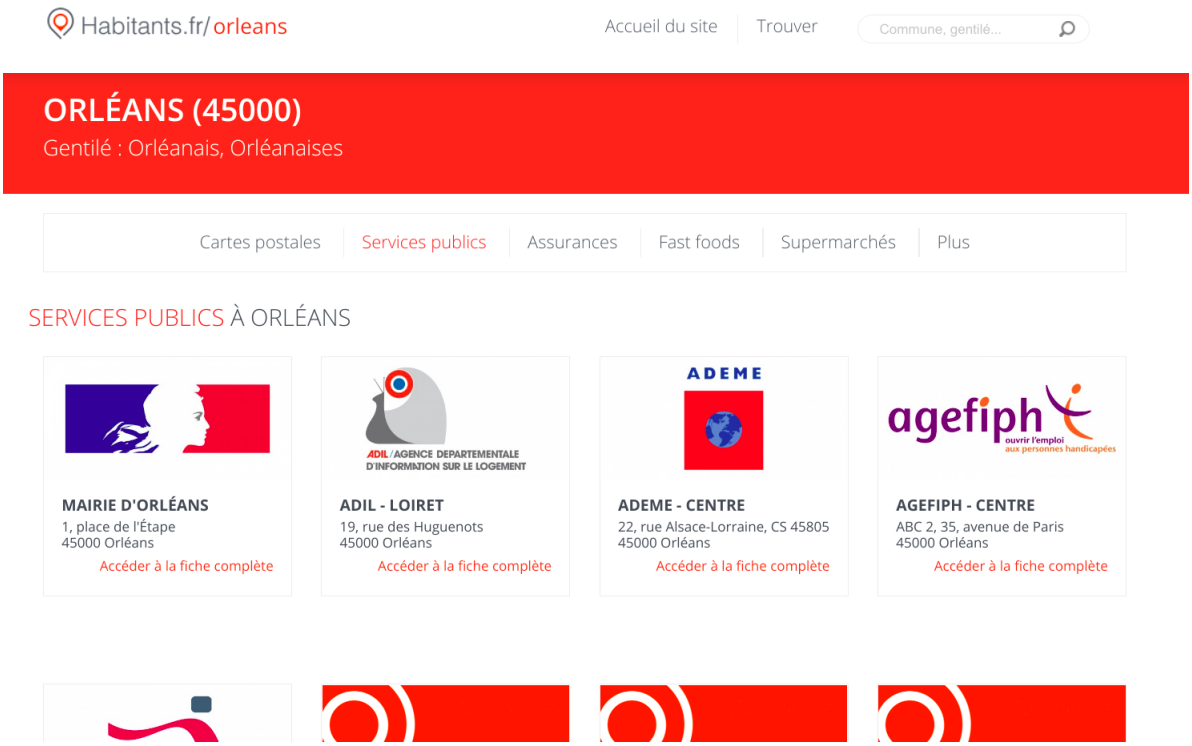

FIGURE 1 – Page web tirée de habitants.fr

.1.2 À partir de Wikipedia

La majeur partie des divisions administratives en France ont une page Wikipedia dédiée mentionnant, pour la plupart, le gentilé de ses habitants. Une solution envisagée est d'utiliser directement une base de connaissance (*knowledge graph*) telle que Wikidata [2] ou DBPedia [3] en essayant de trouver une propriété

1. `https://www.habitants.fr/orleans/services-publics`
2. `https://www.wikidata.org/wiki/Wikidata:Main_Page`
3. `https://www.dbpedia.org/`

ayant la sémantique du gentilé. Nos recherches n'ont pas été concluentes, quelques éléments contiennent bien une propriété **demonym** mais sont minoritaires.

Nous ne passons donc pas par une base de données mais utilisons directement l'HTML des pages Wikipedia. La chaîne de traitement consiste à trouver, en premier lieu, pour un nom de division donné, le ou les éléments Wikidata correspondant. Une simple requête SPARQL (voir figure 2) permet de le faire. L'étape précedente permet de récupérer les identifiants des éléments Wikidata de toutes les divisions dministratives correspondant à un nom.

```
SELECT distinct ?id
WHERE
{
  ?id rdfs:label ?label.
  ?id wdt:P17 wd:Q142. # ie: country:France
  ?id wdt:P1448 ?name.   # ie: official_name:?name
  filter(regex(?name,"^orléans\$", "i")).
  SERVICE wikibase:label { bd:serviceParam wikibase:language "fr" }.

}
```

FIGURE 2 – Requête SPARQL pour récupérer l'ID de l'élément Wikidata correspondant au nom "Orléans"

L'élément Wikidata permet de récupérer l'URL de la page Wikipedia correspondant. La figure 3 montre l'emplacement de l'information que nous cherchons dans la page, qui est récupérée grâce à un outil appelée **wptools**[4] et qui permet d'interroger l'HTML d'une page via son URL et d'en extraire les éléments voulus qui est dans notre cas le gentilé dans l'infobox.

4. https://github.com/siznax/wptools/

FIGURE 3 – Page Wikipedia de la commune d'Orléans. La partie encerclée de l'infobox contient les gentilés de la commune

Bibliographie

[1] Eleni Adamopoulou et Lefteris Moussiades. "An Overview of Chatbot Technology". In : *Artificial Intelligence Applications and Innovations*. Sous la dir. d'Ilias Maglogiannis, Lazaros Iliadis et Elias Pimenidis. Cham : Springer International Publishing, 2020, p. 373-383. isbn : 978-3-030-49186-4.

[2] D. Bobrow et al. "GUS, A Frame-Driven Dialog System". In : *Artif. Intell.* 8 (1977), p. 155-173.

[3] Hongshen Chen et al. "A Survey on Dialogue Systems: Recent Advances and New Frontiers". In : *CoRR* abs/1711.01731 (2017). arXiv : 1711.01731. url : http://arxiv.org/abs/1711.01731.

[4] John J. Godfrey, Edward C. Holliman et Jane McDaniel. "SWITCHBOARD: Telephone Speech Corpus for Research and Development". In : *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*. ICASSP'92. San Francisco, California : IEEE Computer Society, 1992, p. 517-520. isbn : 0780305329.

[5] Daniel Jurafsky et James H. Martin. "Speech and Language Processing". In : Draft of December 30, 2020. Chap. 24 (Chatbots and dialogue systems). url : https://web.stanford.edu/~jurafsky/slp3/24.pdf.

[6] Daniel Jurafsky et James H. Martin. "Speech and Language Processing". In : Draft of December 30, 2020. Chap. 14 (Dependency parsing). url : https://web.stanford.edu/~jurafsky/slp3/14.pdf.

[7] Erdal Kuzey et al. "Temponym Tagging: Temporal Scopes for Textual Phrases". In : *Proceedings of the 6th Temporal Web Analytics Workshop (TempWeb '16)*. ACM, 2016, p. 841-842.

[8] Véronique Moriceau et Xavier Tannier. "French Resources for Extraction and Normalization of Temporal Expressions with HeidelTime". In : mai 2014.

[9] Ashwin Paranjape et al. "Neural Generation Meets Real People: Towards Emotionally Engaging Mixed-Initiative Conversations". In : *CoRR* abs/2008.12348 (2020). arXiv : 2008.12348. url : https://arxiv.org/abs/2008.12348.

[10] James Pustejovsky et al. "TimeML: A Specification Language for Temporal and Event Expressions". In : (jan. 2003).

[11] Minghui Qiu et al. "AliMe Chat: A Sequence to Sequence and Rerank based Chatbot Engine". In : *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada : Association for Computational Linguistics, juil. 2017, p. 498-503. doi : 10.18653/v1/P17-2079. url : https://aclanthology.org/P17-2079.

[12] Kiran Ramesh et al. "A Survey of Design Techniques for Conversational Agents". In : *Information, Communication and Computing Technology*. Sous la dir. de Saroj Kaushik et al. Singapore : Springer Singapore, 2017, p. 336-350. isbn : 978-981-10-6544-6.

[13] Hannah Rashkin et al. "Towards Empathetic Open-domain Conversation Models: A New Benchmark and Dataset". In : *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy : Association for Computational Linguistics, juil. 2019, p. 5370-5381. doi : 10.18653/v1/P19-1534. url : https://aclanthology.org/P19-1534.

[14] Alan Ritter, Colin Cherry et Bill Dolan. "Unsupervised Modeling of Twitter Conversations". In : *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California : Association for Computational Linguistics, juin 2010, p. 172-180. url : https://aclanthology.org/N10-1020.

[15] Stephen ROLLER et al. “Recipes for building an open-domain chatbot”. In : *CoRR* abs/2004.13637 (2020). arXiv : 2004.13637. URL : https://arxiv.org/abs/2004.13637.

[16] Iulian Vlad SERBAN et al. “A Survey of Available Corpora for Building Data-Driven Dialogue Systems”. In : *CoRR* abs/1512.05742 (2015). arXiv : 1512.05742. URL : http://arxiv.org/abs/1512.05742.

[17] Iulian Vlad SERBAN et al. *A Survey of Available Corpora for Building Data-Driven Dialogue Systems*. 2017. arXiv : 1512.05742 [cs.CL].

[18] Lifeng SHANG, Zhengdong LU et Hang LI. “Neural Responding Machine for Short-Text Conversation”. In : *CoRR* abs/1503.02364 (2015). arXiv : 1503.02364. URL : http://arxiv.org/abs/1503.02364.

[19] Yiping SONG et al. “Two are Better than One: An Ensemble of Retrieval- and Generation-Based Dialog Systems”. In : *CoRR* abs/1610.07149 (2016). arXiv : 1610.07149. URL : http://arxiv.org/abs/1610.07149.

[20] Jannik STRÖTGEN et Michael GERTZ. “Multilingual and Cross-domain Temporal Tagging”. In : *Language Resources and Evaluation* 47.2 (2013), p. 269-298. DOI : 10.1007/s10579-012-9179-y.

[21] Oriol VINYALS et Quoc V. LE. “A Neural Conversational Model”. In : *CoRR* abs/1506.05869 (2015). arXiv : 1506.05869. URL : http://arxiv.org/abs/1506.05869.

[22] Wayne WARD et Sunil ISSAR. “Recent Improvements in the CMU Spoken Language Understanding System.” In : jan. 1994. DOI : 10.3115/1075812.1075857.