camp
tocamp

INNOVATIVE SOLUTIONS
BY OPEN SOURCE EXPERTS

LLM et infrastructure de données géographiques

Cas d'utilisation possible dans l'Infrastructure Nationale de Données Géographiques Suisse

1. Contexte

1.1. Introduction

Ce document analyse les applications possibles de “l’intelligence artificielle générative” et plus spécifiquement des grands modèles de langage (“Large Language Model” ou LLM) dans le cadre d’une Infrastructure Nationale de Données Géographiques (INDG).

Ce document est réalisé dans le cadre d’un mandat de conseil donné à Camptocamp par Swisstopo pour répondre au point *4-24-02* du *Plan d’action 2024* de la *Stratégie suisse pour la géoinformation*.

1.2. Besoin et demande

Le point *4-24-02* du *Plan d’action 2024*, met en évidence la difficulté, pour un utilisateur, à trouver l’information qu’il recherche dans une infrastructure de données géographiques. En effet, pour trouver ce qu’il cherche, un utilisateur doit “disposer de très bonnes connaissances du domaine, savoir comment formuler sa demande, connaître les modalités de structuration des géodonnées et être au fait des formes dans lesquelles elles sont disponibles”.

Le besoin exprimé est donc d’évaluer dans quelle mesure les grands modèles de langage peuvent être utilisés pour faciliter l’accès à l’information dans le contexte d’une infrastructure nationale de géodonnées.

Le *Plan d’action 2024* définit le besoin en deux parties:

- La première partie de la demande consiste à explorer comment les LLM peuvent être utilisés pour *améliorer l’expérience d’utilisateurs non experts cherchant des réponses à des questions liées au territoire*.
- La deuxième partie de la demande consiste à *définir la mesure dans laquelle ces LLM peuvent être entraînés à répondre de façon fiable à des questions dont la réponse se trouve dans une infrastructure de géodonnées.*

1.3. Délivrables

Le projet est articulé en trois points:

- Un rapport sur l’état de l’art de l'utilisation de LLM dans le monde des infrastructures de géodonnées et des applications web géospatiales.
- Sur la base de l’état de l’art, une description de ce qui pourrait être fait dans l’infrastructure nationale de géodonnées, en discutant les points suivants:
  - Fiabilité des résultats.
  - Dépendance à des API externes et possibilité du OnPrem et de l’Open Source.
  - Discussion haut niveau sur les coûts
- Un Proof of Concept sur la recherche de jeux de données dans l'infrastructure de géodonnées qui contient plusieurs listes, services ou catalogue de données. L’objectif du PoC est de faciliter la recherche de jeux de données (layers) et non pas de données individuelles (features géographiques).

Un premier rapport répond au premier point. Ce document répond au deuxième point. Un PoC est défini pour le troisième. Des liens sont faits avec le PoC qui démontre certaines propositions décrites dans ce document.

L’ambition de ce document est de permettre au lecteur d’imaginer ce qui est possible à l’aide des LLM dans le contexte d’une infrastructure de géodonnées, de comprendre quels seraient les bénéfices pour les utilisateurs ainsi que de se faire une idée des implications techniques.

2. Introduction

2.1. Qu'est-ce que les LLM peuvent apporter à une plateforme de géodonnées?

Parmi toutes les utilisations possibles des grands modèles de langage (et de leurs petits frères les modèles d'embedding), nous présentons ci-dessous trois catégories de fonctionnalités qui nous semblent avoir le plus de potentiel à court et moyen terme pour l'Infrastructure Nationale de Données Géographiques.

2.1.1. Amélioration de la recherche

La recherche est une des fonctionnalités qui peut le plus facilement bénéficier des apports des modèles de langage. Pour des utilisateurs qui ne connaissent pas le vocabulaire spécifique du domaine, la recherche sémantique, seule ou combinée avec la recherche lexicale et géographique (recherche hybride), est souvent plus performante que la recherche lexicale seule, actuellement mise à disposition.

Un modèle de langage peut aussi être utilisé pour juger et améliorer la pertinence des résultats retournés par un service de recherche; il peut affiner ou reformuler la requête si les résultats ne sont pas pertinents.

L'amélioration de la pertinence de la recherche à l'aide de LLM peut être intégrée dans une application web sans nécessiter une refonte complète de l'expérience utilisateur.

2.1.2. Expérience utilisateur conversationnelle - CUI

A moyen terme, un impact majeur des LLM sur l'informatique en général vient de l'introduction des interfaces utilisateurs conversationnelles ("CUI", *Conversational User Interface*).

Avec une GUI (*Graphical User Interface*) classique, l'utilisateur doit apprendre les fonctionnalités qui lui sont proposées à l'aide d'icônes, de champs de saisie et de menus. L'entrée et la sortie sont généralement fortement structurées.
Avec une CUI, l'utilisateur exprime son besoin dans son langage, par oral ou par écrit. Le programme répond par du texte, une action ou un changement d'état de son interface graphique.

L'utilisation du moyen de communication naturel des humains pour l'interaction homme-machine diminue la courbe d'apprentissage et réduit les barrières d'entrée. Les CUI sont très flexibles et simples d'utilisation mais il est difficile de garantir les résultats en raison de l'ambiguïté du langage naturel.

Les CUI ne vont généralement pas remplacer les GUIs, mais plutôt s'y intégrer et les complémenter. C'est un élément important pour améliorer l'expérience des utilisateurs non experts.

2.1.3. Amélioration et enrichissement des données et métadonnées

Dans l'INDG, des métadonnées de qualité sont le prérequis d'un service de recherche performant.

Les LLM peuvent être utilisés pour améliorer la qualité des métadonnées existantes ou les enrichir en créant des nouveaux champs de recherche basés sur les informations existantes.

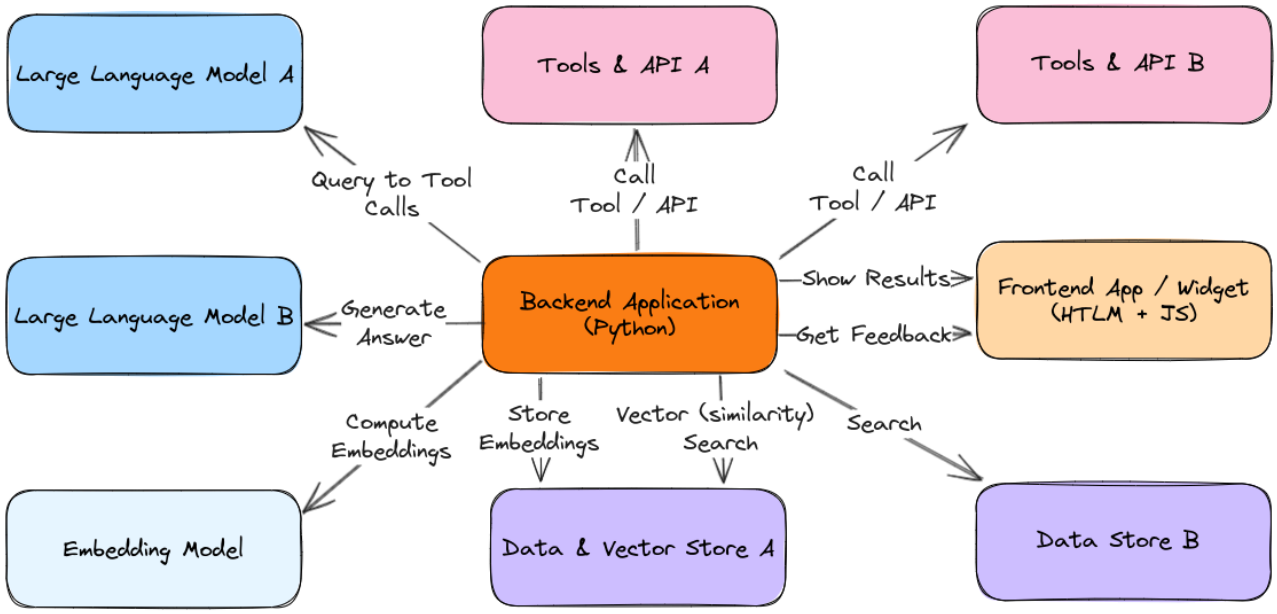
Par exemple, les fonctionnalités suivantes peuvent être implémentées pour améliorer ou enrichir des métadonnées à l'aide d'un LLM:

- ➔ Extraire des informations de classification d'une description textuelle, par exemple pour construire une navigation par facettes ou un système de filtre.
- ➔ Traduire une description d'un langage spécialisé à un langage plus simple et généraliste pour faciliter la recherche pour des utilisateurs non experts.

En règle générale, les résultats obtenus par un projet de *Machine Learning* sont très fortement dépendants de la qualité des données utilisées ("garbage in, garbage out"). Une grosse partie de l'effort est dédiée à l'analyse, la préparation et le nettoyage des données. Ce travail, souvent assez laborieux, peut être facilité par un LLM.

2.2. Anatomie d'une application basée sur les LLM

Comme décrit ci-dessus, les cas d'utilisation des LLM sont très variés. Les systèmes basés sur des LLM diffèrent en fonction des cas d'utilisation qu'ils couvrent. Néanmoins, un certain nombre d'éléments apparaissent fréquemment dans les applications basées sur des LLM. Le schéma ci-dessous montre les éléments courants d'une application de type *Retrieval Augmented Generation*.



Éléments courants dans une application Retrieval Augmented Generation.

2.2.1. Modèles

En dehors du très connu ChatGPT, il existe des milliers de modèles de langage, généralistes ou spécialisés pour résoudre une tâche précise.

Il n'est pas rare qu'un système utilise plusieurs modèles. Par exemple, on peut imaginer une application qui utilise un modèle d'embedding pour la recherche sémantique, un LLM spécialisé et peu coûteux pour identifier les appels à des outils et un modèle généraliste et coûteux pour générer une réponse textuelle.

2.2.2. Data Stores

Les techniques du RAG (cf. premier rapport) et de la recherche sémantique sont centrales dans beaucoup d'applications qui utilisent des LLM. Il est donc courant qu'un tel système comprenne ou

puisse accéder à plusieurs dépôts de données, généralement avec la possibilité de faire des recherches vectorielles et hybrides.

Il existe beaucoup de nouveaux data stores spécialisés dans la recherche vectorielle. Dans le cadre du POC et de ce document, nous nous concentrerons sur l'utilisation des data stores qui sont déjà en place dans la INDG, à savoir Elasticsearch et PostgreSQL, avec les extensions qui fournissent le support de la recherche sémantique.

2.2.3. Outils et API

Pour répondre aux requêtes des utilisateurs, il est souvent nécessaire de faire appel à des outils ou des données externes au système. Il est relativement commun qu'une application RAG utilise des sources externes comme wikipedia, yahoo finance ou google ou des sources plus spécialisées.

Dans le cadre d'une application géospatiale, on peut par exemple imaginer que le système puisse accéder à un modèle d'élévation au travers d'une API.

2.2.4. Applications

Le backend interagit avec les datastores, les outils, les API et les modèles afin de répondre aux requêtes des utilisateurs. C'est le cœur du système où il est possible d'injecter la connaissance du domaine spécifique pour lequel l'application est développée.

Le frontend met à disposition de l'utilisateur les fonctionnalités du système, généralement au travers d'une interface conversationnelle (CUI) ingérée dans une GUI.

Le frontend d'une application de *machine learning* permet généralement à l'utilisateur d'exprimer un jugement sur la qualité de la réponse fournie. Collecter les retours des utilisateurs est une condition nécessaire à la mise en place indispensable de processus d'amélioration continue (*active learning*).

2.3. Quels critères pour décider de l'utilisation d'un LLM? Par ou commencer?

Avec l'engouement actuel autour des LLM et de l'IA générative, il est tentant de vouloir appliquer ces nouvelles technologies à tous les problèmes que nous rencontrons. Comment fixer ses priorités? Cette section décrit brièvement quelques critères à prendre en compte pour choisir les domaines qui bénéficieront le plus de l'utilisation de l'IA générative et des LLM.

2.3.1. Copilote ou agent autonome?

Pour commencer, notons qu'il y a différentes façons d'utiliser les LLM et d'envisager l'interaction entre l'homme et la machine:

- ➔ Le mode **autonome**, dans lequel le système à base de LLM remplace un autre système autonome ou le travail d'un humain.
- ➔ Le mode **copilote**, dans lequel le LLM aide un humain à faire son travail, mais c'est au final l'humain qui valide et prend la responsabilité du résultat.

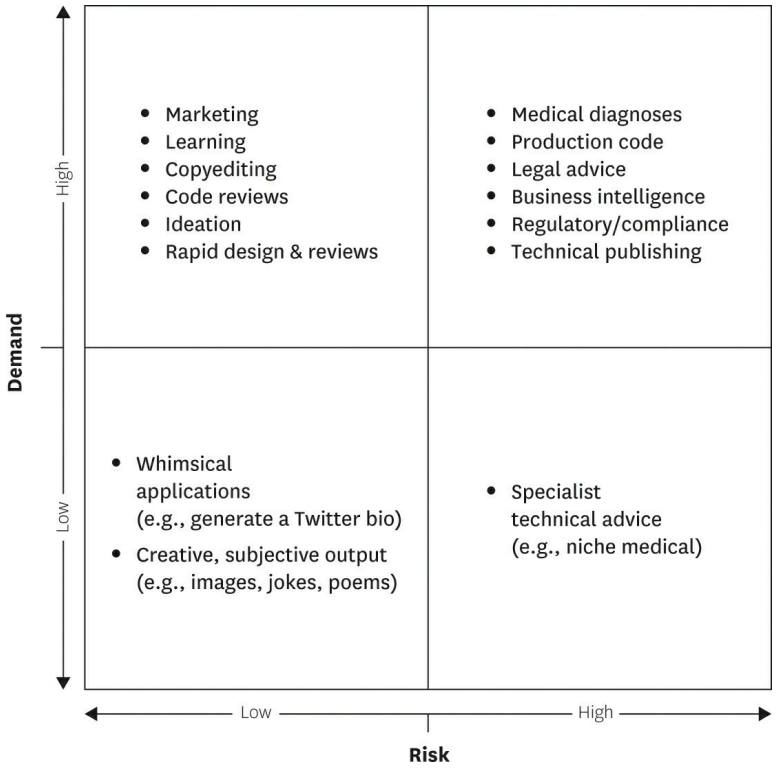
Lors de la prise de décision sur l'utilisation ou non d'un LLM pour résoudre un problème précis, il faut prendre en compte que le LLM peut être utilisé soit de manière autonome, soit comme un assistant pour un humain.

2.3.2. Matrice Demande / Risque

Un critère nécessaire est la demande ou la valeur pour les utilisateurs finaux. Plus la demande est grande, plus l'incitation à réaliser une fonctionnalité est importante.

Un autre critère important se situe au niveau du risque lié aux réponses inexactes que pourrait fournir le système. Quelles sont les conséquences d'une erreur?

Le rapport de la Harvard Business Review propose de visualiser la demande et le risque dans une matrice 2×2 et d'**expérimenter l'introduction de l'IA Générative sur des projets qui sont dans le quadrant supérieur gauche, ou la demande est forte et le risque faible**.



Pour en savoir plus:

- [Harvard Business Review Report on Generative AI]

2.3.3. Les questions à poser

Lors du choix d'un projet en relation avec les LLM, il faut impérativement se poser les questions suivantes:

- Va-t-on résoudre un problème qui impacte nos utilisateurs?
- Est-ce que les LLM sont bons pour résoudre ce problème?
- Ce problème est-il fréquent? Est-ce qu'il y a un sens économique à automatiser sa résolution?
- Quelle est la conséquence d'une erreur?

Lorsque la conséquence d’une erreur est importante, cela ne signifie pas forcément que l’on ne peut pas utiliser de LLM. Par contre, on préférera se diriger vers une solution de type copilote, avec un humain qui valide (ou pas) les propositions du système.

3. Améliorer la recherche à l'aide des LLM

Un des cas les plus évidents d'utilisation des LLM concerne l'amélioration des fonctionnalités de recherche. C'est un cas intéressant car il permet de se familiariser avec l'utilisation et l'intégration des LLM sans les exposer directement à l'utilisateur final. Améliorer la recherche à l'aide de LLM ne nécessite pas la mise en place d'une interface utilisateur conversationnelle, cela peut-être fait sans changer l'expérience utilisateur.

La première partie de cette section décrit les techniques qui peuvent être utilisées pour améliorer la recherche. La seconde partie décrit des cas d'utilisation concrets qui pourraient être appliqués à la NDG et illustre certains points en faisant des liens avec le POC.

Comme l'amélioration de la recherche sur les métadonnées est le sujet du Proof of Concept, ce chapitre est plus détaillé que ceux sur les interfaces conversationnelles ou l'amélioration des métadonnées.

3.1. Techniques d'amélioration de la recherche à l'aide des LLM

3.1.1. Introduction

Partons d'une recherche lexicale classique, basée sur la correspondance des mots de la requête de l'utilisateur avec ceux extraits des documents. Elle est généralement basée sur des indexes inversés qui prennent en compte la fréquence des mots et évalue l'importance d'un terme contenu dans un document, relativement à une collection de document (TF-DF pour "*term-frequency-inverse document frequency*", cf. wikipedia). A partir de là, il est possible d'utiliser des modèles de langages de différentes manières pour améliorer la pertinence des résultats par rapport à la recherche lexicale actuelle que nous considérons comme notre point de comparaison, notre *baseline*.

Les techniques suivantes peuvent être utilisées et sont décrites dans cette section:

- **Création de synonymes**: Pour augmenter les chances d'une correspondance entre les termes de la requête de l'utilisateur et ceux indexés à partir des documents de référence, il est possible d'utiliser un LLM pour créer des synonymes. Ceci peut être fait à l'indexation ou à l'exécution de la requête ou les deux.
- **Reformulation de la requête**: Un LLM peut être utilisé pour reformuler la requête de l'utilisateur afin qu'elle fournisse de meilleurs résultats. La reformulation peut comprendre l'extraction d'un élément particulier, comme un lieu géographique, afin de l'utiliser comme un filtre.
- **Recherche sémantique:** Certains modèles de langage peuvent être utilisés pour calculer des représentations vectorielles de textes et utiliser ces représentations pour faire une recherche basée sur le sens du texte (cf premier rapport).
- **Mixer les techniques et merger les résultats**: Les recherches lexicales et sémantiques ont chacune leurs forces et leurs faiblesses. Les meilleurs résultats sont généralement obtenus en combinant plusieurs techniques de recherche dans ce que l'on appelle la *recherche hybride*.
- **Réordonner et filtrer les résultats** ("ReRanking"): Les modèles de langages peuvent être utilisés pour filtrer et réordonner les résultats fournis par le service de recherche. Les LLM sont assez performants pour juger de la pertinence d'un document pour répondre à une requête.

Les techniques ci-dessus peuvent être combinées. Chacune de ces méthodes a un coût, mesuré en temps de travail, en argent dépensé pour utiliser des LLM et en latence pour l'utilisateur final. Pour juger de l'opportunité d'implémenter et de maintenir une ou plusieurs des techniques décrites ci-dessus, il faut pouvoir mesurer le bénéfice qu'elles apportent et comparer leurs rapports coûts /

bénéfices. Il est donc indispensable de pouvoir mesurer le bénéfice apporté par ces techniques. **Il faut pouvoir quantifier la performance d'un service de recherche pour l'améliorer.**

3.1.2. Création de synonymes

Une difficulté couramment rencontrée dans l'implémentation d'un service de recherche est la différence de vocabulaire entre les documents de références, rédigés par des connaisseurs du domaine, et les requêtes des utilisateurs qui souvent ne sont pas des spécialistes.

Pour mitiger cette difficulté, il est possible d'utiliser un LLM pour faire converger le langage des documents et de la requête. Ceci peut être fait de deux manières:

- **A l'indexation**, on demande à un LLM de générer un résumé ou des mots-clés pour chaque document à indexer. Ces mots-clés (ou ce résumé) sont des synonymes des mots-clés utilisés pour décrire le document, mais dans un vocabulaire plus proche de celui utilisé par les clients cibles de l'application. En indexant des synonymes, on augmente les chances qu'une requête lexicale soit couronnée de succès, indépendamment des potentielles différences de vocabulaire. Cette méthode étant appliquée à l'indexation des documents, elle n'a pas d'impact sur la latence ou les coûts par requête.
- **A la requête**, il est possible de reformuler la requête de l'utilisateur à l'aide d'un LLM pour que le vocabulaire de cette dernière soit plus proche de celui utilisé dans les documents indexés. Même sans prendre en compte le vocabulaire potentiellement particulier des documents indexés, le fait d'ajouter des synonymes aux requêtes des utilisateurs augmente les chances qu'une requête lexicale soit couronnée de succès. Cette méthode étant appliquée lors de la réponse à une requête, elle a un impact sur le temps de réponse et les coûts car elle implique un appel à un LLM.

Ces deux techniques, l'augmentation des données indexées ou de la requête par des synonymes, ne sont pas exclusives et peuvent être combinées.

3.1.3. Reformulation de la requête

Il existe de nombreuses techniques basées sur des LLM pour transformer la requête d'un utilisateur afin qu'elle soit plus performante ou qu'elle utilise au mieux les fonctionnalités et la syntaxe du service de recherche. D'autre part, il est courant d'injecter de la connaissance métier pour améliorer la requête de l'utilisateur (avec ou sans l'aide d'un LLM).

Voici quelques catégories de transformation d'une requête utilisateur:

- **Extraction / construction d'un filtre**: Il est parfois possible d'extraire un filtre de la requête de l'utilisateur et de l'utiliser pour améliorer la qualité des résultats.
  Par exemple, un service de recherche géospatial permettra souvent de contraindre une recherche textuelle à une zone géographique. La requête "les pharmacies à Lausanne" devrait donc être transformée en une requête qui recherche "pharmacie" avec un filtre géographique qui correspond à l'empreinte géographique de la ville de Lausanne. Ce filtre géographique est construit à partir de la requête de l'utilisateur (et parfois de services externes, par exemple pour obtenir la géométrie de la ville de Lausanne).
- **Ajout d'informations implicites**: Il est parfois possible d'ajouter des filtres qui ne sont pas explicitement exprimés dans la requête de l'utilisateur, mais sont déduits du contexte ou de l'état de l'application (sans forcément avoir besoin de LLM pour cela).
  Par exemple, il est possible d'ajouter la langue comme critère de recherche en la déduisant de la langue sélectionnée par l'utilisateur pour l'interface graphique ou en utilisant un modèle pour la déterminer. On peut aussi imaginer restreindre certaines requêtes à certaines sources de données uniquement (confédération, cantons, communes par ex) car on sait qu'un type de

donnée est gérée par une catégorie de fournisseur de données. On injecte de la connaissance métier dans la requête de l'utilisateur.

- **Construction d'une requête structurée**: un cas particulier de la reformulation, poussée à l'extrême, sont les modèles de type Text2SQL. Dans ce cas, un modèle est entraîné à transformer une requête en langage humain en une requête en langage informatique comme SQL. Il existe beaucoup d'exemples intéressants de Text2SQL, mais sa mise en pratique est compliquée, SQL étant un langage généraliste et bas niveau. Il est plus aisé de générer des requêtes structurées pour des langages plus restreints et haut niveau comme OGC API ou Web Feature Service (WFS).

La liste ci-dessus est loin d'être exhaustive. Il existe beaucoup de manières de transformer une requête utilisateur pour la rendre plus performante. L'ajout de synonymes dont nous avons parlé ci-dessus en est aussi une. Notons que la formulation optimale pour une requête dépend du service de recherche. Il y a donc un couplage fort entre la transformation de la requête et la recherche à proprement parler.

Parmis les techniques de transformation de requêtes, on peut aussi citer:

- *Rewrite-Retrieve-Read* est une technique qui consiste à demander à un LLM de reformuler la requête de l'utilisateur. Décrite dans le papier *Query Rewriting for Retrieval-Augmented Large Language Models* publié en octobre 2023.
- *Step back prompting* est une technique qui essaie de généraliser la question de l'utilisateur pour la rendre moins spécifique. Il est possible que l'utilisateur pose une question plus spécifique que la description présente dans les métadonnées. La question plus générique est parfois combinée avec la question spécifique pour la phase de *retrieval*. Publiée dans *Take a step back: evoking reasoning via abstraction in large language model*, paru en mars 2024.
- *RAG-Fusion* génère *n* requêtes à partir de la requête de l'utilisateur et recherche les documents qui s'y rapportent. Décrite dans *A New Take on Retrieval-Augmented Generation*, paru en février 2024.
- *Hypothetical Document Embeddings* (HyDE) est une autre façon de contourner le problème de la différence entre la requête de l'utilisateur et les documents de référence. Cette technique consiste à générer, à partir de la requête utilisateur, un document hypothétique qui a les caractéristiques des documents de référence (taille, forme, ton). On utilise ensuite ce document dans la phase de *retrieval* pour rechercher les documents similaires. Cette technique est décrite dans *Precise Zero-Shot Dense Retrieval without Relevance Labels*, paru en décembre 2022.

3.1.4. Recherche sémantique

La recherche sémantique consiste à utiliser un modèle dérivé d'un modèle de langage pour créer une représentation vectorielle d'un texte. A chaque texte correspond un vecteur (appelé *embedding*) qui représente sa signification. Les textes dont les vecteurs sont proches du vecteur qui représente la requête de l'utilisateur ont des significations similaires à la requête et sont retournés comme résultats (cf premier rapport pour plus de détails).

En principe, la recherche sémantique est robuste face aux variations de vocabulaire et même de langue, pour autant que les langues utilisées soient toutes supportées par le modèle d'*embeddings*.

Notons que les vecteurs d'*embeddings* ont une taille fixe. Pour des performances optimales, les textes recherchés et retournés par une telle recherche doivent être approximativement de même taille; il est donc nécessaire de découper les documents volumineux.

Si le nombre de documents indexés est important, il n'est plus possible de calculer la distance entre la requête et tous les documents pour trouver les plus proches (*nearest neighbor)*. Il est donc nécessaire d'utiliser un algorithme de *approximate nearest neighbor* (comme Hierarchical navigable small world) qui sacrifient un peu de précision contre des meilleurs temps de réponse.

Un moyen très efficace pour améliorer la qualité des résultats d'une recherche vectorielle est **d'adapter le modèle d'*embedding***, utilisé pour générer les vecteurs qui représentent le texte, à son jeu de données. Affiner (*fine-tuner*) un modèle d'*embeddings* est réalisable à un coût acceptable, ces modèles étant d'une taille qui permet de les affiner sur une seule machine avec un accélérateur graphique (GPU).

La difficulté réside dans la construction d'un jeu de données pour affiner le modèle. Ce jeu de données est une liste de triplet: requête, document positif, document négatif. Il est généralement constitué à l'aide des retours des utilisateurs, dans une démarche d'active learning (voir ci-dessous).

Le *fine-tuning* de modèles d'*embedding* est une technique intéressante et efficace pour adapter un modèle à un jeu de données. Il n'est pas nécessaire d'entraîner à nouveau le modèle à chaque modification des données.

3.1.5. Recherche hybride

Nous avons parlé de recherche lexicale, de recherche sémantique et de filtre géographique. Les retours d'expérience montrent que les meilleures performances en termes de qualité de résultat sont généralement obtenues en combinant plusieurs types de recherches différentes.

Comment combiner des résultats de recherche?

Il existe plusieurs méthodes:

- **Filtre booléen**: Il est possible d'ajouter un filtre booléen sur une recherche lexicale ou géospatiale. Ce filtre peut-être textuel, sur un attribut ou géographique.
  Par exemple, il est possible de rechercher les jeux de données qui parlent de vélo dont le propriétaire est la Confédération; ou alors, les jeux de données qui parlent de vélo dans la commune de Morges.
- **Requêtes multiples**: Les recherches sémantiques et lexicales retournent des listes ordonnées de résultats avec un score. Elles ne peuvent pas être combinées simplement, les scores entre différents types de recherche n'étant pas simplement comparables. Dans ce cas, il faut trouver un moyen de créer une liste de résultats consolidés à partir des résultats de chaque recherche. L'approche naïve consiste à prendre les N premiers résultats de chaque type de recherche. Les approches plus complexes impliquent l'utilisation d'un LLM pour sélectionner les résultats les plus pertinents.

Dans le cas courant des requêtes multiples qui nécessitent de fusionner les résultats, on peut aussi en profiter pour les trier et les ré-ordonner, comme décrit ci-dessous.

3.1.6. Réordonner et filtrer les résultats

Les modèles de langages sont performants lorsqu'il s'agit de juger si un texte donné est pertinent pour répondre à une question. On peut donc les utiliser pour juger de la qualité des résultats, supprimer ceux qui ne sont pas pertinents et ordonner ceux qui restent.

Cette étape, appelée "*reranking*" est appliquée à la fin du processus. Elle peut aussi être utilisée pour juger de la qualité des résultats et prendre des actions correctives s'ils ne sont pas bons (comme par exemple retravailler la requête initiale de l'utilisateur et recommencer le processus).

3.2. Évaluer et mesurer pour améliorer la recherche

Nous avons vu plusieurs méthodes pour améliorer la qualité d'un service de recherche. Toutes ont des bénéfices théoriques et des coûts pratiques en termes de complexité, de latence ou d'argent.

Comment peut-on décider lesquels appliquer? Où se trouvent les rapports coûts / bénéfices les plus intéressants?

Pour répondre à cette question, il faut pouvoir mesurer la qualité des résultats retournés et la mettre en rapport avec les coûts qui sont mesurés en termes de complexité, augmentation des temps de réponse et coûts financiers liés à l'utilisation d'un LLM.

Pour mesurer la qualité des résultats, il faut construire un jeu de données de test avec les propriétés suivantes:

- **Forte corrélation avec l'objectif**: Le jeu de test doit représenter fidèlement l'objectif à atteindre. Concrètement, pour un service de recherche, une amélioration des résultats sur le jeu de test doit se traduire par une amélioration de la qualité de la recherche, tel que la perçoit l'utilisateur. De même, une détérioration des résultats sur le jeu de test doit se traduire par une détérioration de la qualité perçue par l'utilisateur.
- **Taille suffisante**: Le jeu de données de test est utilisé pour comparer différentes alternatives et améliorer incrémentalement les performances du service de recherche. Il faut donc un nombre d'exemples assez important pour pouvoir mesurer et comparer les performances des différentes solutions, avec une granularité suffisante.

Les résultats sur le jeu de test permettent de mesurer l'évolution de la qualité du service de recherche. Pour juger de la pertinence d'une solution, il faut aussi prendre en compte d'autres éléments, pas directement liés aux réponses fournies par le service. Ces éléments sont:

- **La latence**: l'ajout d'appels en série à des LLM pour améliorer les résultats de la recherche entraîne une augmentation des temps de réponses qui peuvent devenir inacceptables. Une bonne pratique consiste à fixer un plafond sous lequel le temps de réponse doit être maintenu.
- **Les coûts**: utiliser un LLM coûte de l'argent. Une requête de recherche qui fait 2 ou 3 appels à des LLM aura un coût d'opération significativement plus élevé qu'une recherche lexicale standard. Ce paramètre est très dynamique car les prix des LLM évoluent (à la baisse) en permanence.
- **La complexité**: Est-ce que écrire et maintenir des centaines de lignes de code pour gagner 2% de précision à du sens? La complexité est difficile à mesurer, mais elle doit aussi être prise en compte lors du choix de la solution, car elle implique des coûts de maintenance à moyen terme.

Évaluer, quantifier les coûts et les bénéfices est relativement complexe et consommateur de temps mais absolument indispensable. C'est un sujet important et souvent sous-estimé dans les projets de machine learning.

3.3. Recherche sur les métadonnées

Une application possible de la recherche améliorée par LLM dans le cadre de l'INDG est la recherche sur les métadonnées. L'utilisateur qui cherche la réponse à une question sur le territoire doit commencer par identifier le jeu de donnée (le "layer") qui lui permet de répondre à sa question.

La recherche sur les métadonnées est le sujet d'un Proof of Concept réalisé en parallèle avec ce rapport. Ce sujet est donc significativement plus détaillé dans ce document, ce qui permet d'expliquer ce qui a été réalisé concrètement dans le cadre du PoC.

3.3.1. Problématique

Le site map.geo.admin.ch contient plusieurs centaines de jeux de données géographiques. Le catalogue geocat.ch contient la description de plusieurs milliers de jeux de données géographiques, dont ceux de map.geo.admin.ch mais aussi ceux d'une partie des cantons et communes de Suisse.

Certaines descriptions de couches géographiques (les métadonnées) sont très complètes et contiennent une empreinte géographique alors que d'autres sont beaucoup plus sommaires.

Bien que map.geo.admin.ch et geocat.ch fournissent des interfaces de recherche textuelle par mot clé, il n'est pas toujours facile pour l'utilisateur d'identifier le jeu de données qui lui permet de répondre à sa question. La difficulté à trouver le jeu de données adéquat est due à:

- La multiplicité des sources d'information: le grand public connaît peut-être map.geo.admin.ch, mais pas forcément geocat.ch. Il aura donc de la peine à trouver le jeu de données qui permet de répondre à sa question si ce dernier n'est pas publié sur map.geo.admin.ch.
  Savoir **où chercher** est une première difficulté.
- L'utilisation de recherche lexicale: si l'utilisateur n'a pas le même vocabulaire que les gestionnaires de données, alors il ne trouvera pas ce qu'il cherche.
  Savoir **quels termes utiliser** est une seconde difficulté.
- La description des jeux de données est très variable d'un fournisseur à l'autre. Certains jeux de données sont décrits dans plusieurs sources.
  La **qualité des métadonnées** est aussi une difficulté.

Pour ces raisons, il n'est pas toujours évident pour un utilisateur peu au fait des pratiques de la gestion des géodonnées de trouver l'information qu'il recherche, surtout si cette dernière n'est pas publiée sur map.geo.admin.ch.

3.3.2. Pistes d'amélioration

Pour faciliter la recherche de jeux de données géographiques, nous voyons trois axes d'améliorations potentielles basées sur les LLM:

- **Recherche améliorée à l'aide de modèles de langage:**
  Implémentation de la recherche améliorée à l'aide de LLM, décrite ci-dessus.
- **Conversational UX**:
  Mise en place d'une interface conversationnelle qui guide l'utilisateur dans sa recherche.
- **Amélioration des métadonnées**
  Utilisation d'un LLM pour juger de la qualité des métadonnées existantes et les améliorer.

Cette section est dédiée à l'utilisation de LLM pour améliorer la recherche proprement dite. C'est aussi le sujet du PoC. L'utilisation de LLM dans le cadre d'interfaces conversationnelles ou l'amélioration des métadonnées grâce à un modèle de langage sont traités plus loin dans le document.

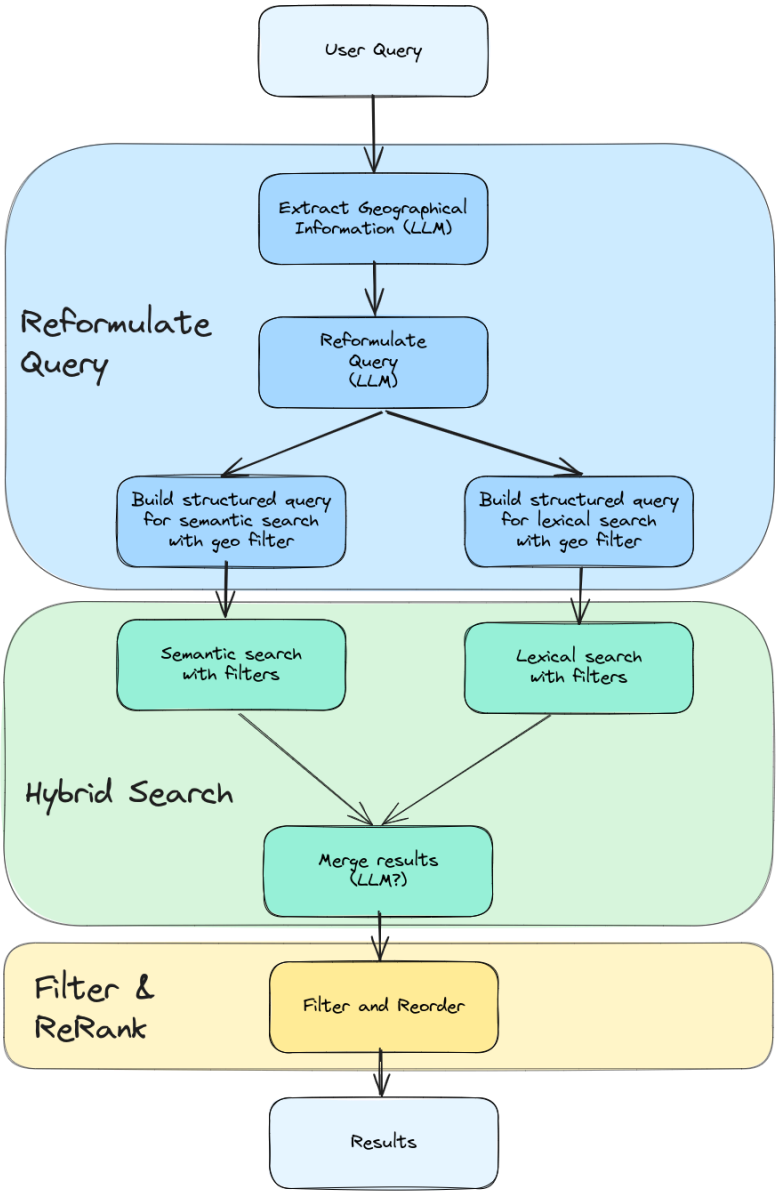
3.3.3. Amélioration de la recherche sur les métadonnées à l'aide de modèles de langage

3.3.3.1. Vue d'ensemble

L'amélioration de la recherche sur les métadonnées est basée sur les éléments suivants:

1. Développement d'une méthode d'évaluation des performances du service de recherche
2. Reformulation de la requête de l'utilisateur et extraction des lieux géographiques
3. Recherche sémantique sur la description des couches géographiques
4. Filtres géographiques pour contraindre la recherche sémantique
5. Recherche hybride en ajoutant la recherche lexicale existante de geocat.ch
6. Filtre et tri des résultats de la recherche hybride par un LLM
7. Mise en place des processus et outils nécessaires à l'amélioration continue ("active learning").

Certains points critiques sont développés ci-dessous.



Exemple de recherche géospatiale basée sur des LLM

3.3.3.2. Méthode d'évaluation des performances du service de recherche

Pour évaluer les performances d'un service de recherche, il faut commencer par construire un jeu de données de test. Ce jeu de test est composé de paires (requête utilisateur, réponse attendue).

Les requêtes utilisateurs peuvent par exemple être en partie extraites des logs des services de recherche de geocat.chou de geoadmin, ce qui permet d'avoir des demandes réelles, faites par des utilisateurs.
Les réponses attendues sont soit un document de métadonnées, soit un groupe de documents si la réponse à une requête demande la consultation de plusieurs jeux de données.

Le jeu de données de test devrait être composé au minimum d'une centaine d'exemples de bonne qualité, représentatifs de l'utilisation du service (en termes de thèmes, langue, longueur des requêtes, vocabulaire utilisé, etc.).

Une fois le jeu de test construit, une fonction d'évaluation des performances du service de recherche est sélectionnée. Cette fonction prend en entrée le jeu de test et un service de recherche et retourne un score entre 0 et 1. Zéro signifie que le service de recherche ne retourne jamais la réponse attendue, 1 signifie que le service de recherche retourne toujours la réponse attendue.

Si chaque requête ne doit retourner qu'un seul jeu de données, alors la fonction d'évaluation est simplement le rapport entre le nombre d'exemples correctement prédit et le nombre total d'exemples (*accuracy*).

Cette étape laborieuse de construction d'un jeu de test et de sélection d'une méthode d'évaluation des performances prend du temps et demande des connaissances du domaine. Une méthode d'évaluation pertinente donne des résultats fortement corrélés avec la satisfaction des utilisateurs; il faut donc connaître et expliciter les besoins de ces derniers. Pour cela, des techniques en provenance de la création de produit peuvent être utilisées (analyse des interactions des utilisateurs avec l'application, segmentation des utilisateurs, interview et enquêtes auprès des utilisateurs). La création de jeux de test et de méthodes d'évaluation de qualités est indispensable et déterminante pour la qualité du projet dans son ensemble. Dans un projet de développement d'une application RAG ou d'un service de recherche (sémantique), les différents tests et évaluations (pour chaque étape et de bout en bout) représentent une part plus importante de l'effort que dans un projet de développement classique (pour lequel le standard est d'environ 20%).

3.3.3.3. Eléments techniques

Afin d'implémenter la recherche sémantique, il faut une base de donnée vectorielle et un modèle d'embedding.

PostgreSQL et l'extension PGVector est une possibilité intéressante, étant donné que PosgreSQL est déjà abondamment utilisé par l'INDG. C'est le choix qui a été fait pour le PoC.

Il existe un grand nombre de modèles d'embedding, à opérer soi-même ou accessible sous forme de service (Software as a Service, SaaS).

- Dans un premier temps, il est plus simple d'utiliser un modèle en SaaS, comme `text-embedding-3-small` ou `text-embedding-3-large` d'OpenAI. C'est le choix qui a été fait pour le PoC.
- Dans un deuxième temps, il est possible de changer pour un modèle open source, hébergé dans l'infrastructure locale ou en SaaS. Le jeu de test permet de mesurer les différences de performances sur ce cas d'utilisation entre les différents modèles.

Si la description d'un jeu de données a changé, sa représentation vectorielle doit être mise à jour. Ceci implique la mise en place d'un processus de synchronisation entre les sources de données et la base vectorielle.

Le choix de l'information (des champs) à inclure pour la recherche sémantique peut avoir un impact important sur les performances. Il est donc pertinent de tester différentes alternatives.

3.3.3.4. Filtre géographique et qualité des données

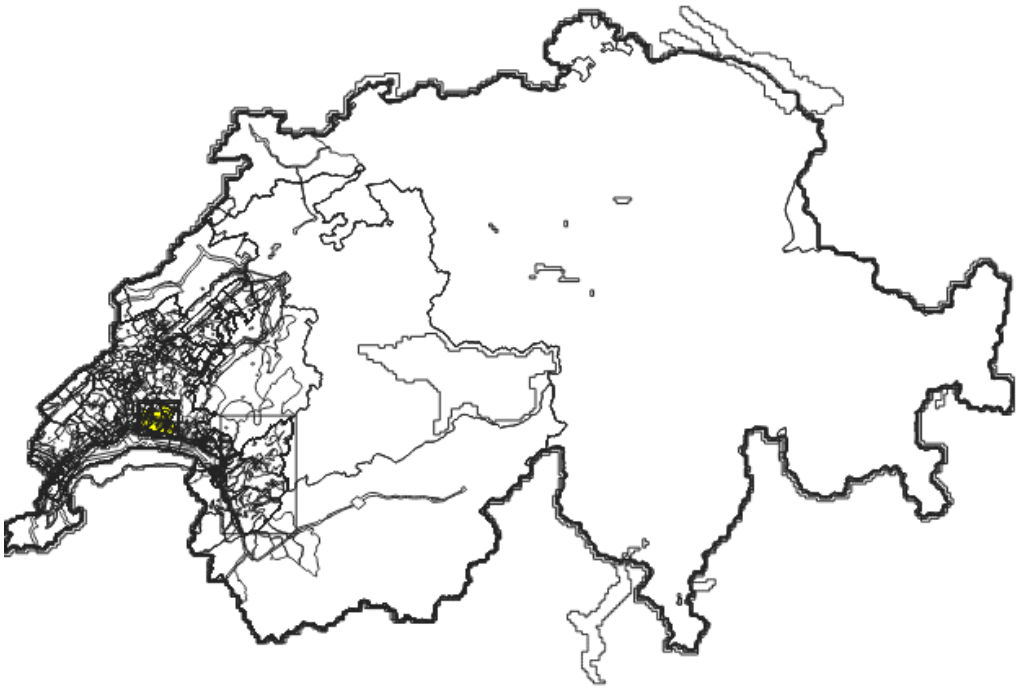
Pour la requête "Schwimmbad in Thun", le système ne doit pas retourner une couche du canton de Vaud, même si elle traite de piscines et bains.

A l'aide du lieu géographique extrait de la requête, il est possible de sélectionner les jeux de données qui couvrent ou intersectent la localisation extraite de la requête.

Deux difficultés sont à prendre en compte:

- Dans geocat.ch, seule une partie des layers ont des polygones associés.
  Cette difficulté peut être mitigée soit en complétant les métadonnées dans geocat, soit en utilisant les bbox (qui peuvent poser des problèmes en raison de l'imprécision géographique importante qu'elles introduisent).

➔ Les BD vectorielles ne supportent pas nativement les requêtes géographiques.
Cette difficulté peut être contournée en se limitant aux outils qui supportent à la fois les requêtes géographiques et la vectorielle, comme PostgreSQL avec PostGIS et PGVector.



Couverture géographique des layers avec des géométries précise dans geocat

Certains jeux de données n'ont pas de géométrie précise associée, mais ont une *bounding box*. Cette dernière pourrait être prise en compte au cas où il n'y a pas de géométrie précise associée au jeu de données. La meilleure solution consiste à améliorer la qualité des métadonnées, en ajoutant des polygones (manuellement ou automatiquement lorsque cela est possible).

3.3.3.5. Amélioration continue - Active learning

Pour les projets de machine learning, la première mise en production est souvent considérée comme la moitié du projet. En effet, une application de *machine learning* (ML) doit être utilisée pour être améliorée. Aucun jeu de données de test créé manuellement ne sera meilleur que le retour des utilisateurs.

D'une certaine manière, l'active learning pour un projet de ML a des similarités avec le développement agile pour un projet de développement classique, puisqu'elle consiste en une amélioration continue et itérative qui prend en compte les retours des utilisateurs.

Pour mettre en place l'amélioration continue (*active learning*), il est nécessaire de collecter le feedback des utilisateurs, le reviewer et en faire un jeu de données. Ceci nécessite la mise en place d'éléments d'interface graphique (thumb up/down), de stockage et d'un outil pour revoir les retours des utilisateurs et les intégrer dans le jeu de test.

Les processus de collecte des retours, de test des nouvelles versions et de comparaison de la version n avec la version n-1 doivent être automatisés afin de rendre l'amélioration continue efficace.

Ces automatisations, généralement regroupées sous le terme de *MLOps,* ne sont pas détaillées dans ce document.

3.3.3.6. Opportunités

Cette section fournit un survole des risques et des coûts d'un tel projet afin de le placer dans la matrice d'opportunité de la Harvard Business Review, présentée au début de ce document.

Risques

L'utilisation de la recherche sémantique n'implique pas de génération de texte par un modèle de langage. Il n'y a donc pas de risque d'hallucination.

Le risque de fournir des **résultats de recherche inexacts** existe, mais n'est pas différent du risque existant avec d'autres techniques de recherche comme la recherche lexicale. Il est possible d'écrire des jeux de tests pour un service de recherche.

Les risques liés aux **changements d'expérience utilisateur** sont aussi faibles, la recherche sémantique ou hybride pouvant remplacer la recherche lexicale sans changement d'interface utilisateur significatif.

Globalement, les risques associés à la mise en place de la recherche sémantique sont les mêmes que ceux liés à la mise en place d'un service de recherche lexicale. Le risque principal est probablement celui lié à **la valeur et au besoin**: l'amélioration de la recherche apporte-elle une vraie valeur à l'utilisateur? Répond-elle à un besoin? Ses bénéfices sont-ils supérieurs aux coûts?
Une manière de mitiger ce risque est **d'analyser les logs du service de recherche de jeux de données** qui sont aujourd'hui en production. En construisant un ensemble de tests basés sur les logs du service de recherche de production, on s'assure que les tests correspondent à des demandes réelles des utilisateurs et on peut comparer les résultats de la recherche actuelle avec les résultats de la recherche sémantique ou hybride. Il est aussi possible d'estimer la performance de la recherche actuellement disponible.

Coûts

Notons les points suivants en relation avec les coûts de la mise en place d'une solution qui utilise la recherche sémantique:

- Les coûts de fonctionnement des modèles d'embedding, en SaaS ou déployé dans sa propre infrastructure sont très significativement inférieurs aux coûts de fonctionnement d'un modèle de langage. Par exemple, chez OpenAI (23.10.2024):
  - Modèle de langage gpt-4o: 2.5$ / M input tokens, 10$ / M output tokens
  - Modèle de langage gpt-4o-mini: 0.150 / M input token, 0.6$ / M output tokens
  - Modèle d'embedding text-embedding-3-small: 0.02$ / M input tokens
  - Modèle d'embedding text-embedding-3-large: 0.13$ / M input tokens
- Le travail nécessaire pour intégrer la recherche sémantique au niveau du frontend et de l'interface utilisateur est faible; la recherche sémantique présente une expérience utilisateur similaire à la recherche lexicale à la différence prêt que les utilisateurs peuvent (et sont encouragés à) introduire des questions au lieu de mots-clés dans le champ de recherche.
- Il ne faut pas négliger le temps de travail nécessaire pour **augmenter la qualité** et passer d'un PoC à une application en production. Faire une bonne démo est souvent possible, atteindre le niveau requis pour de la production est un défi majeur des projets de ML.
  En particulier, il faut prévoir un effort important pour:
  - Écrire un grand nombre de tests réalistes, par exemple en utilisant les logs de la recherche.
  - Définir et implémenter des moyens de test et de comparaison de deux versions du système.
  - Implémenter et tester les améliorations itératives.
  - Implémenter un processus de feedback qui permette de collecter les retours des utilisateurs.
  - Utiliser efficacement les retours des utilisateurs.

- Pour être efficaces, l'amélioration continue ("active learning") ainsi que l'affinage (fine-tuning) des modèles d'embeddings (si implémentés) doivent être automatisés.

La mise en place de la recherche sémantique peut être faite comme un remplacement d'un service de recherche existant et implique moins de risques et de coûts que d'autres fonctionnalités liées aux LLM, comme le développement et le déploiement d'interfaces utilisateur conversationnels.

3.3.3.7. Améliorations à tester

Pour aller plus loin, il serait intéressant d'effectuer des tests dans les domaines suivants:

- Modèles d'embedding: il existe un grand nombre de modèles d'embeddings, en SaaS ou Open Source. Ces différents modèles peuvent avoir un impact important en termes de performances et de coûts.
- Affinage (*fine-tuning*) de modèle d'embedding.
- Impact du multilinguisme: les représentations vectorielles des métadonnées sont en théorie en grande partie indépendantes de la langue. Il faudrait mesurer la différence de performance si on utilise le même embedding pour toutes les langues ou si on calcule les embeddings et fait une recherche spécifique pour chaque langue.

Ces améliorations potentielles ne pourront pas être testées en détail dans le cadre du Proof of Concept.

3.4. Recherche sur les données géographiques

Une autre application possible des modèles de langage est liée à la recherche sur les données proprement dite. Une fois que le jeu de données est identifié, l'étape suivante est de trouver les éléments du jeu de données qui permettent de répondre à la question de l'utilisateur.

3.4.1. Problématique

Lorsqu'un utilisateur a une question sur le territoire, Il peut être intéressé par un jeu de données ("les chemin de randonnée autour de Wabern") ou par une information précise, généralement contenue dans les attributs d'une entrée faisant partie d'un jeu de données ("quelle est la surface du bâtiment de la Seftigenstrasse 264 à Wabern ?").

Dans le deuxième cas, la réponse attendue n'est pas un jeu de données géographique, mais elle est contenue dans les attributs d'une ou plusieurs entrées d'un jeu de données.

Une fois le jeu de données identifié, comment peut-on aider l'utilisateur à trouver le ou les objets qui l'intéressent ?

L'INDG propose des API qui permettent de faire des recherches sur les attributs des objets géographiques (service Find) ou pour obtenir les objets contenus dans une surface (service identify). Ces services sont destinés aux développeurs d'applications et ne sont pas prévus pour être appelés directement par des utilisateurs finaux. Ils sont utilisés pour des sites comme map.geo.admin.ch.

En règle générale, un utilisateur qui a une question dont la réponse se trouve dans l'attribut d'un objet doit identifier le jeu de donnée adéquat, faire une recherche géographique pour trouver l'objet dans ce jeu de données et finalement afficher ses attributs. Il s'agit d'un processus en plusieurs étapes qui demande une certaine maîtrise des outils à disposition. Les modèles de langages peuvent-ils participer à la simplification de ce processus afin de le rendre accessible à un plus grand nombre ?

3.4.2. Pistes d'amélioration

Les données géospatiales peuvent être très hétérogènes et les LLM ne vont pas apporter une solution miracle qui s'applique à tous les cas. Néanmoins, au moins deux pistes intéressantes peuvent être explorées:

- **Recherche basée sur les LLM pour les attributs textuels**: Si un jeu de données est composé d'attributs textuels, il est possible d'utiliser une recherche sémantique sur ces attributs ainsi que les différentes techniques décrites dans le chapitre sur la recherche sur les métadonnées.
- **Appel de fonction par un LLM**: Un modèle de langage peut être utilisé pour appeler des fonctions (informatique) en réponse à des requêtes en langage naturel. Cette capacité permet de répondre à des cas d'utilisation courants identifiés. Par exemple, si on constate que beaucoup d'utilisateurs s'intéressent à des informations sur les bâtiments, on peut écrire une fonction *getBuildingInfo(adresse)* qui utilise les API de geo.admin.ch. Un LLM est utilisé pour appeler cette fonction quand l'utilisateur pose une question sur un bâtiment.

Les possibilités d'amélioration de la recherche à l'aide de modèles de langage ont été amplement discutées dans le cadre de la recherche sur les métadonnées.

La possibilité d'appeler des fonctions en réponse à des requêtes utilisateurs est développée et exemplifiée dans le chapitre sur les interfaces conversationnelles.

4. Expérience utilisateur conversationnelle - CUI

Les LLM ouvrent de nouvelles perspectives en termes d'expérience utilisateur. C'est un des domaines où ils auront le plus d'impact à moyen-long terme. En effet, il est aujourd'hui possible de remplacer les clicks qu'un utilisateur doit faire pour expliquer à une application ce qu'il veut par une conversation entre l'utilisateur et l'application. Les *expériences utilisateur conversationnelles* (*Conversational User Interface, CUI*) vont s'imposer comme un élément obligatoire d'une expérience utilisateur moderne.

L'introduction du langage comme moyen de communication homme-machine peut être faite à différents niveaux, en plusieurs étapes, avec des complexités variables.

- La **recherche** peut-être vue comme le premier niveau: l'utilisateur n'entre plus des mots-clés dans un champ de recherche; il peut écrire une question telle qu'il se la pose et un modèle de langage va transformer sa question en une requête destinée à un moteur de recherche (sémantique, lexical ou hybride).
- Le **réponse aux questions** (*question answering*) est une évolution de la recherche dans laquelle au lieu de retourner les documents de référence, un LLM extrait de ces documents la réponse à la question de l'utilisateur.
- Le **dialogue**, dans lequel l'utilisateur peut poser des questions qui font références aux interactions précédentes dans la conversation, est une évolution de la réponse aux questions.
- Il y a différents niveaux de dialogue, avec des niveaux de complexité variés. Un chatbot peut, par exemple, **demander des précisions à l'utilisateur** afin de collecter les informations manquantes pour répondre à sa requête.
- Finalement, un LLM peut être utilisé pour **appeler des fonctions** prédéfinies (*function / tool calling)*. Dans ce cas, le modèle traduit du langage naturel en appel de fonction. Il est ainsi possible de mettre simplement à disposition des fonctionnalités complexes.

L'interface utilisateur conversationnelle est basée sur le principe du RAG (Retrieval Augmented Generation), décrit dans le premier rapport, qui permet de baser la conversation sur des informations pertinentes et à jour.

4.1. Techniques des interfaces conversationnelles

4.1.1. Répondre à des questions (RAG Question Answering)

La réponse à des questions basées sur les données (ou métadonnées) présentes dans l'INDG est une évolution naturelle de la recherche sémantique. En effet, la fonctionnalité de réponse aux questions se base sur la recherche sémantique ou mixte pour sélectionner les documents desquels elle va extraire la réponse à la question de l'utilisateur.

Contrairement à l'expérience utilisateur de la recherche sémantique (une interface de recherche classique, avec une liste de résultats), la fonctionnalité de réponse aux questions ne fournit généralement pas une liste de résultats, mais uniquement une réponse textuelle à la question de l'utilisateur avec un lien sur le document duquel est extraite la réponse.

La réponse aux questions peut être intégrée dans une interface de recherche standard, sans nécessiter le développement d'une expérience utilisateur conversationnelle. C'est ce que propose actuellement Google dans sa liste de résultats de recherche.

Dans le contexte géospatial, la réponse aux questions pourrait être une carte au lieu d'une réponse textuelle. Si l'utilisateur demande "Où se trouvent les zones inondables dans le canton de Berne?", l'application répond avec une carte centrée et zoomée sur le canton de Berne et avec le layer adéquat sélectionné. La différence avec la recherche sémantique est que l'on ne retourne pas une liste de

résultats, mais une carte pré-configurée (layer, centre, zoom) pour répondre à la question de l'utilisateur.

4.1.2. Dialoguer avec l'utilisateur - RAG Chatbots

La possibilité de dialoguer de manière naturelle est ce qui a fait le succès fulgurant de ChatGPT et a ouvert la voie aux expériences utilisateurs conversationnelles.
La possibilité d'utiliser des informations en provenance de ses propres données rend ce nouveau type d'expérience utilisateur applicable à des plateformes spécialisées comme celles de l'INDG.

La réponse aux questions est une évolution naturelle de la recherche sémantique, dans laquelle on extrait pour l'utilisateur la réponse qu'il recherche au lieu de lui fournir les documents dans lequel se trouve cette réponse. Les chatbots sont le pas suivant; on permet à l'utilisateur d'interagir avec le système comme s'il parlait avec un humain qui l'aide à résoudre son problème. L'utilisateur peut demander des précisions ou rebondir sur une réponse (*follow up question*). Il a une interaction naturelle car il n'a pas à apprendre à utiliser le système, c'est ce dernier qui parle sa langue, idéalement.

Implémenter un chatbot de qualité est complexe car l'espace des entrées possibles est pratiquement infini. Il faut gérer l'historique de la conversation, les possibles sous-entendus ainsi que les ambiguïtés qui caractérisent le langage humain. Il faut aussi que les réponses fournies soient basées sur des données exactes et à jour. C'est l'objectif d'un *Chatbot RAG* (*Retrieval Augmented Generation*), un chatbot qui converse avec l'utilisateur et base ses réponses sur les informations contenues dans la base de connaissance de l'application.

Les chatbots existent depuis longtemps, mais les LLM leur donnent des capacités impossibles à atteindre avec les technologies précédentes. C'est LA technologie à la mode que beaucoup d'organisations veulent intégrer à leur application. Néanmoins, atteindre un niveau de qualité satisfaisant reste un défi majeur et beaucoup de chatbots actuels génèrent plus de frustration que de satisfaction.

Ci-dessous, différents types de chatbots sont décrits afin de donner une idée des fonctionnements possibles de ces interfaces conversationnelles.

4.1.2.1. Chatbot RAG statique

Un chatbot statique est un chatbot dont le flux d'exécution est pré-déterminé (sous forme de graph acyclique). Généralement, son fonctionnement ressemble beaucoup à ce qui est fait pour le *question answering*:

1. L'utilisateur pose une question, qui peut référencer des éléments de dialogue précédent (Dans *quels cantons suisse parlent-on français? <réponse bot> Et lequel est le plus grand?*)
2. Le chatbot RAG utilise un LLM pour reformuler la dernière question afin qu'elle soit indépendante
   (→ *Quel est le plus grand canton suisse dans lequel on parle français?*)
3. Le chatbot RAG fait une recherche par similarité, avec la question reformulée, pour trouver les documents qui contiennent la réponse.
4. Le chatbot RAG utilise un LLM pour extraire la réponse des documents retournés par la recherche par similarité.

D'un point de vue fonctionnel, la principale différence avec la réponse aux questions est la gestion du contexte et de l'historique de la conversation. Ceci comprend la reformulation de la question pour qu'elle soit indépendante du contexte et la gestion de l'historique de la conversation.

Les étapes du processus sont toujours les mêmes. Le temps d'exécution est prévisible et relativement constant. C'est ce type de chatbot qui sont utilisés pour le support; ils fournissent une interface conversationnelle à des pages de documentation ou de FAQ.

4.1.2.2. Chatbot dynamique - Agent

Un agent est un programme dont le flux d'exécution est en partie déterminé par un modèle de langage. Suivant la question, le chatbot peut décider de répondre directement, de faire une recherche sémantique ou d'utiliser un autre outil mis à sa disposition.

L'avantage des chatbot agentiques est qu'ils sont plus souples et peuvent répondre à un éventail plus large de demande. L'inconvénient est qu'ils sont plus difficiles à tester et que leur flux d'exécution, et donc leur temps de réponse, n'est pas connu à l'avance.

Notons que le terme "agent" est un buzzword à la mode qui a plusieurs définitions et dont la signification change avec le temps. L'utilisation d'agents est aussi connue sous le terme de Agentic AI.

4.1.2.3. ReAct: Reasoning and Acting Agent

Un agent ReAct est un cas particulier simple d'un agent, décrit dans un papier publié en mars 2023 et appelé ReAct: Synergizing Reasoning and Acting in Language Models. Il utilise la capacité qu'ont les LLM récents de retourner une réponse structurée ou un appel de fonction à la place du texte (décrite en détail en §4.1.3).

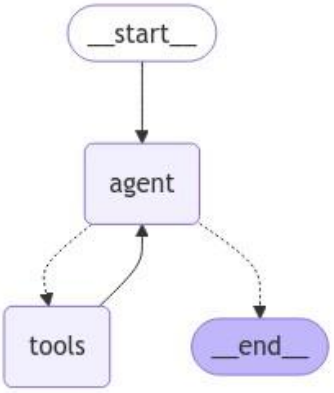
Dans l'architecture ReAct, pour "Reasoning and Acting" le système a un certain nombre d'outils à sa disposition et décide s'il peut répondre à la demande de l'utilisateur directement ou s'il doit faire appel à un outil car il n'a pas encore la réponse. Les outils peuvent offrir la possibilité d'appeler des API (recherche wikipedia ou google, API d'un service météo, etc.), d'appeler une fonction locale (recherche dans un service interne, recherche sémantique) ou même de demander des précisions à l'utilisateur.

Le flux d'exécution d'un agent ReAct est décrit ci-dessous.

> Etant donné un ensemble d'outils `[tools]` utilisable par le système et une requête `req`:
>
> 1. *Reason*: Le LLM estime-t-il qu'il peut répondre à la `req` sans informations complémentaires?
>   a. OUI: répond et termine.
>   b. NON: sélectionne un outil (parmi les `[tools]`) à appeler pour obtenir des compléments d'information.
> 2. *Act*: Appel le tool sélectionné avec les paramètres adéquats, ajoute son output à `req` et retourne à 1.

Avec l'algorithme ci-dessus, il n'est pas possible de savoir à l'avance combien d'étapes (de boucles "*Reason-Act*") seront nécessaires avant que la réponse ne soit retournée (on peut bien entendu configurer un maximum). C'est le LLM qui détermine s'il doit appeler un outil ou retourner la réponse à l'utilisateur. Le temps de réponse n'est pas prévisible, même si le nombre d'itérations peut être borné.

Pour éviter un temps d'attente long et inexpliqué, il est possible d'informer l'utilisateur du raisonnement de l'agent au cours de son exécution, notamment grâce à la fonctionnalité streaming des requêtes HTTP (la réponse arrive au fil de l'eau). Ceci donne un retour constant sur le fait que l'agent fait quelque chose, rend l'échange dynamique et évite la confusion et la frustration dans l'esprit de l'utilisateur qui peut s'impatienter et se demander si la réponse va finir par arriver.



Graph d'exécution d'un agent ReAct. *(img source: langgraph.com)*

Dans le graph ci-dessus, le noeud "agent" représente les appels au LLM qui décide d'utiliser un outil (noeud *tools*) ou de répondre (noeud *_end_*).

Ce pattern est un point de départ très utilisé pour développer un chatbot agentic.

4.1.3. Outils et appels de fonctions par des LLM

Les LLM ont la capacité d'appeler des fonctions prédéfinies pour répondre à une requête utilisateur. Cette fonctionnalité a été initialement proposée par ChatGPT (cf. documentation), mais elle est aujourd'hui disponible dans beaucoup de modèles, closed et open source. C'est grâce à cette fonctionnalité que l'on peut implémenter aisément le pattern ReAct décrit ci-dessus.

Le principe est relativement simple: On fournit au LLM, avec la requête de l'utilisateur, une liste de signatures de fonctions (nom, paramètres et documentation qui explique à quoi sert la fonction et ses paramètres). Le LLM retourne un ou plusieurs appels de fonction qui correspondent à la demande de l'utilisateur.

Par exemple:

1. On met à disposition du modèle les 2 fonctions suivantes:
   - get_height(lat: float, long: float): float: retourne l'altitude à un point défini par les paramètres lat et long.
   - get_pos(name): retourne la latitude et la longitude
2. On soumet au modèle la requête "Où se trouve Wabern?"
3. Le modèle répond: get_pos(name=Wabern)

Dans l'exemple ci-dessus, le LLM choisit soit d'appeler une des fonctions mises à sa disposition, soit de répondre par une réponse textuelle. Le modèle sélectionne la fonction, mais ne l'exécute pas (il ne reçoit que sa signature et pas son code).

A partir de là, il est possible d'exécuter la fonction et de retourner le résultat à l'utilisateur. Ou de fournir le résultat de l'exécution au modèle pour qu'il génère une réponse finale textuelle ou appelle une autre fonction.

On peut voir l'appel de fonction par un modèle de langage comme la traduction d'une demande du langage humain dans un langage informatique.

4.2. Applications pour l'INDG

4.2.1. Vision

L'introduction d'une interface conversationnelle dans les applications proposées par l'INDG permettrait d'améliorer l'expérience d'utilisateurs non experts, conformément aux objectifs décrits dans le *Plan d'action 2024*, mais son implémentation représente un défi important.

L'objectif à long terme, la vision, est que l'utilisateur puisse simplement converser avec l'application et obtenir les réponses à ses questions sur le territoire. Les questions peuvent être posées par oral ou par écrit et les réponses sont des textes, des cartes ou les deux. Il n'y a pratiquement plus de GUI (Graphical User Interface) car il n'y en a plus besoin, l'application comprend son utilisateur au lieu que ça soit l'inverse.

Cette vision de l'ordinateur qui converse en langage humain avec ses utilisateurs est présente dans de nombreux ouvrages de littérature d'anticipation (science fiction). Elle sera probablement réalisable dans quelques années. Les LLM sont la brique qui manquait jusqu'à récemment. Le défi consiste à trouver le chemin qui permet d'y arriver et à mettre en place les étapes intermédiaires nécessaires à sa réalisation. Ces étapes doivent permettre de construire l'interface conversationnelle de demain en mettant en place les éléments techniques et humains nécessaires à sa réalisation.

Au niveau technique, la CUI idéale nécessite des services de données adaptés, une interface de chatbot, la possibilité de collecter les retours des utilisateurs, des modèles affinés sur les données propres, etc. alors qu'au niveau humain il faut des compétences différentes de celles nécessaires au développement d'une application classique.

Définir des étapes réalistes entre la situation actuelle et l'interface conversationnelle de demain est le défi d'aujourd'hui.

4.2.2. Stratégie

On peut découper le chemin entre l'objectif final et la situation actuelle en commençant par implémenter une CUI sur un scope beaucoup plus réduit. Ceci permet de réduire la complexité et d'augmenter progressivement les compétences à la compréhension de ce nouveau paradigme d'expérience utilisateur.

L'INDG propose des données territoriales sur un nombre très important de sujets: l'environnement, l'économie, les loisirs, les transports et bien d'autres encore. Les premiers objectifs pourraient être définis autours du développement d'une CUI qui se concentre sur un seul de ces sujets.

Une CUI n'est pas magique. Comme une GUI, elle est développée pour répondre à des cas d'utilisation qui doivent être clairement définis et que l'on peut tester et évaluer. Limiter le nombre de cas d'utilisation auxquels doit répondre le chatbot simplifie son développement et permet d'avancer progressivement.

Comme étapes, on peut imaginer le développement d'un chatbot qui couvre un des domaines (thèmes) présents dans les données de l'INGD. Par exemple:

- → Une CUI pour les questions en relation avec l'environnement (BAFU).
- → Une CUI qui répond aux questions sur les loisirs.
- → Une CUI qui répond aux questions sur l'agriculture

En limitant le scope, on limite à la fois le nombre de cas d'utilisation (de catégories de questions) et le nombre de sources de données qui peuvent potentiellement être utilisées pour y répondre. On augmente ainsi les chances de développer une CUI utilisable et engranger de l'expérience.

Ces CUI sont en pratique des chatbots augmentés, car ils peuvent répondre soit par du texte, soit par des cartes. Une fois que plusieurs de ces CUI sont fonctionnelles et satisfaisantes, elles peuvent être fusionnées pour créer une interface conversationnelle qui s'approche de la vision à long terme.

4.2.3. Définir des cas d'utilisation

Qu'entend-on par cas d'utilisation dans le contexte de la vision proposée dans ce chapitre?

Un cas d'utilisation est une catégorie de questions. Cette catégorie est définie par un certain nombre de critères qui permettent de la restreindre suffisamment pour que l'on puisse la tester extensivement. Quelques exemples de critères::

- L'utilisateur cible (ex: randonneur, propriétaire, automobiliste, agriculteur, ...).
- Le ou les jeux de données impliqués.
- Le type de réponse attendu (un texte, une position, une carte pré-configurée, un lien).
- Les outils (API) potentiellement utilisés.
- Un nombre important d'exemples de requêtes.

Le type de réponse varie en fonction de la catégorie de questions. Par exemple:

- Une carte sera la meilleure réponse pour des questions comme:
  - Y a-t-il des chiens de bergers à la Pierre du Moëllé?
  - Où puis-je recharger ma voiture électrique à Bienne?
  - Où se trouvent les héliports en Suisse?
- Une réponse textuelle sera plus adaptée pour des questions comme:
  - Est-ce que Berne est plus près de Lausanne ou de Zurich?
  - Quelle montagne est la plus haute, le Mont Rose ou le Cervin?

Parfois, il est nécessaire d'utiliser des outils ou API et pas uniquement une carte. Par exemple:

- Quelle est l'altitude du stade de la Pontaise à Lausanne? (modèle d'élévation)
- Est-ce que Berne est plus près de Lausanne ou de Zurich? (position et calcule de distance)
- Quelle montagne est la plus haute, le Mont Rose ou le Cervin? (modèle d'élévation, comparaison)
- Show me a map of Paradeplatz in Zürich (carte avec object mis en évidence)

Définir ces cas est un travail en soi. Pour définir ces cas d'utilisation, il faut apprendre à connaître les utilisateurs actuels ou potentiels des services de géodonnées pour comprendre leurs besoins et donc ce qui aurait de la valeur à être proposé au travers d'une interface conversationnelle. Souvent, un chatbot sera surtout utile s'il est capable d'utiliser plusieurs sources d'information de provenance différentes (potentiellement aussi hors de l'INDG).

Le message à retenir est qu'il n'est pas réaliste de vouloir implémenter une interface conversationnelle qui permet de répondre à toutes les questions sur le territoire, du moins dans un premier temps. Il faut définir des cas d'utilisation concrets et sélectionner ceux qui sont réalisables et ont de la valeur pour leur public cible.

4.3. Champ d'application et découvrabilité

Lorsqu'un utilisateur est confronté à une interface conversationnelle, il ne sait pas quelles sont les fonctionnalités mises à disposition au travers de cette interface. Avec une GUI classique, il est possible de parcourir les menus et de survoler les icônes pour se faire une idée des fonctionnalités proposées par l'application. Ce n'est pas le cas avec une interface basée sur le dialogue.

Il faut donc clairement indiquer quelles fonctionnalités sont couvertes et lesquelles ne le sont pas. Ceci peut-être fait dans et hors de la CUI.

Si l'on sort de ces cas d'utilisations définis et testés, on prend le risque que les réponses fournies ne soient plus basées sur des données vérifiées et maîtrisées. Les réponses peuvent être incorrectes soit parce qu'elles sont basées sur des informations peu fiables ou périmées (les données utilisées pour l'entraînement du modèle), soit parce que le modèle de langage "hallucine" car il ne connaît pas la réponse. Il faut donc mettre en place des mécanismes qui limitent ce risque.

4.4. Exemples et illustrations

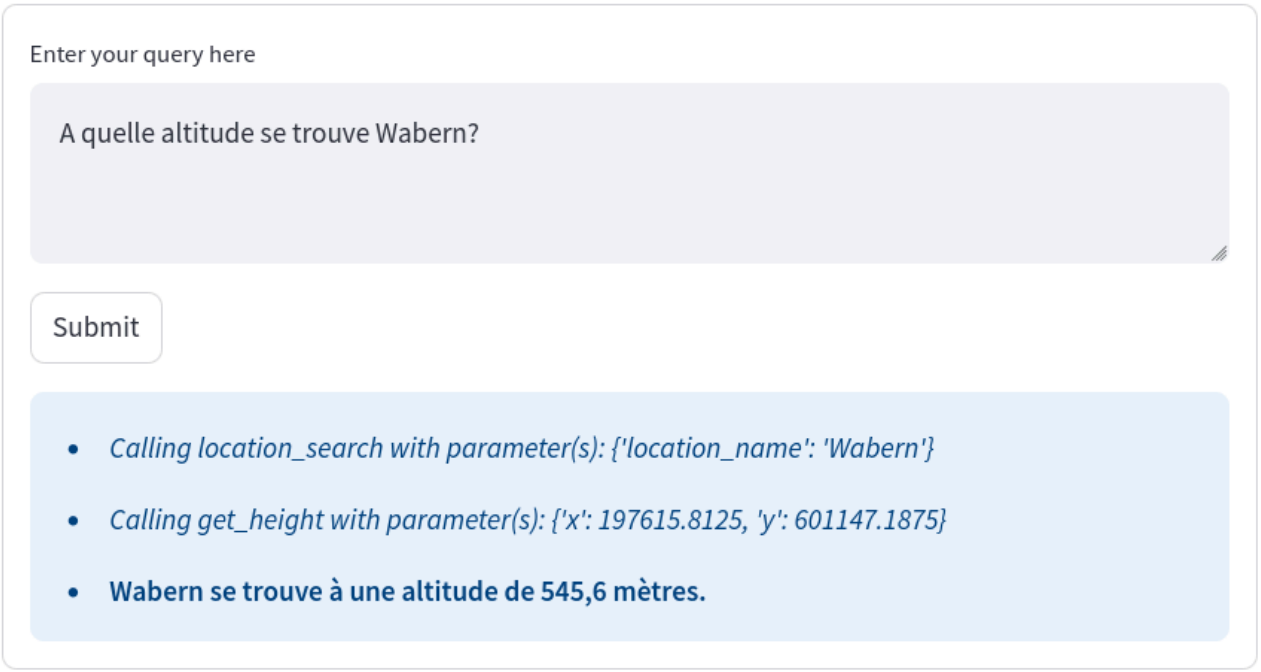
Les exemples ci-dessous illustrent le fonctionnement d'un agent ReAct qui accède aux API de geo.admin.ch:

- `location_search(location_name)` qui prend en paramètre un nom et retourne les coordonnées (long, lat) en utilisant l'API Search de geo.admin.ch.
- `get_height(x,y)` qui prend en paramètre des coordonnées (long, lat) et retourne une altitude en interogeant le modèle d'élévation à l'aide de l'API Height de geo.admin.ch.
- `get_street_map(street_name, community_name)` qui prend en paramètre un nom de rue et un nom de commune et retourne un lien sur une carte centrée sur la rue, avec cette dernière en surbrillance en utilisant l'API Find de geo.admin.ch.
- `compute_distance(x1,y1,x2,y2)` qui prend en paramètre deux paires de coordonnées et calcule la distance euclidienne entre les deux points.

A l'aide de ces quatre fonctions nous avons construit un agent ReAct (décrit ci-dessus) qui peut répondre à des questions des utilisateurs, sans avoir programmé statiquement la suite d'actions à effectuer pour obtenir la réponse.

Ci-dessous, quelques captures d'écran montrent l'exécution de l'agent pour différentes requêtes.

L'agent appelle une fonction qui utilise l'API find puis construit une url vers une carte.



Exemple d'exécution d'un agent ReAct qui enchaîne 2 appels à l'API pour répondre à la demande

Notons à nouveau que pour l'exemple ci-dessus, le code de l'application ne dit pas que pour trouver l'altitude d'un lieu désigné par son nom il faut d'abord trouver sa position. C'est le modèle de langage qui décide de faire ces deux appels à la suite pour répondre à la requête de l'utilisateur.

L'exemple suivant, ci-dessous, implique l'utilisation successive de cinq outils pour répondre à la requête de l'utilisateur.

Enter your query here

Is Bern closer to Lausanne or Zurich?

Submit

- *Calling location_search with parameter(s): {'location_name': 'Bern'}*
- *Calling location_search with parameter(s): {'location_name': 'Lausanne'}*
- *Calling location_search with parameter(s): {'location_name': 'Zurich'}*
- *Calling compute_distance with parameter(s): {'x1': 200386.703125, 'y1': 598633.6875, 'x2': 153352.859375, 'y2': 538612.4375}*
- *Calling compute_distance with parameter(s): {'x1': 200386.703125, 'y1': 598633.6875, 'x2': 247945.34375, 'y2': 682216.0}*
- **Bern is closer to Lausanne, with a distance of approximately 76,254 meters, compared to a distance of about 96,166 meters to Zurich.**

Exemple d'agent ReAct qui accède à l'API Search de l'INDG et à un outil local de calcul de distance.

En se référant au graph, le dernier exemple fait cinq fois la boucle agent-tools avant de générer la réponse (et donc fait 5 appels au LLM, ce qui introduit une certaine latence).

5. Améliorer la qualité des données

5.1. Introduction

Les modèles de langages peuvent être utilisés pour améliorer la qualité ou compléter des données existantes. Il est aussi possible de générer des nouvelles données à base de données existantes.

Dans ce contexte, les capacités des LLM peuvent être utilisée pour :

- Mesurer la qualité d'un jeu de données existant.
- Classer ou sélectionner des mots-clés pour une entrée d'un jeu de données.
- Générer une description textuelle à partir d'information structurée.
- Extraire des informations structurées d'une description textuelle.
- Traduire automatiquement.
- Résumer.
- Dédupliquer en utilisant la capacité des LLM à faire du "fuzzy match".

Automatiser la détection de problèmes de qualité des données, et, dans une certaine mesure, leur correction est précieux, ces tâches étant chronophages et laborieuses.

5.2. Techniques d'amélioration

Cette section développe quelques points abordés dans l'introduction pour améliorer la qualité des données à l'aide de LLM.

5.2.1. Mesurer la qualité d'un jeu de données

Un modèle de langage peut être utilisé pour juger (et noter) la cohérence d'un jeu de données. A l'aide du *prompt* adéquat, un LLM pourra juger des critères comme:

- Est-ce qu'un texte correspond à un standard (défini par des exemples) en termes de niveau de détail, ton, type de vocabulaire utilisé, etc.
- Est-ce qu'une classification (p. ex. des mots-clés) est cohérente avec le contenu d'autres champs textuels.
- Ya-t-il des incohérences entre deux champs (un résumé et une description complète, ou un texte original et une traduction).

Avec un *prompt* assez évolué contenant des exemples de ce qui est attendu, un LLM peut être utilisé pour quantifier la qualité d'un jeu de données et identifier les entrées problématiques.

5.2.2. Générer du contenu structuré à partir de texte

A partir d'une description textuelle, un modèle de langage peut être utilisé pour sélectionner des mots-clés dans un thésaurus. Beaucoup de standards de données contiennent des thésaurus ou des ontologies. Associer les bons termes à un élément qui possède déjà une description peut être potentiellement automatisé à l'aide d'un modèle de langage.

La donnée structurée enrichit le jeu de données initial et permet d'offrir par exemple de nouvelle fonctions de recherche de type toggle ou facettes.

Par exemple: "extraire d'une description d'itinéraire à vélo de Schweizmobil s'il faut porter ou non le vélo". On vient ajouter un attribut booléen au jeu de données, pour offrir une option à la GUI lors de la recherche, un toggle "portage de vélo, oui - non".

5.2.3. Générer du contenu à partir de contenu existant

A l'inverse du point précédent, il est aussi possible de générer ou enrichir une description à l'aide d'un modèle de langage. Ce dernier se basera sur les champs existant (mots-clés par exemple) et des exemples de descriptions pour générer un texte.

Cette utilisation est plus complexe que le cas inverse parce qu'il n'y a pas de solution unique.

5.2.4. Résumer et traduire

La traduction est un des cas d'utilisation les plus courants des modèles de langage, et la qualité proposée est en constante progression. Bien souvent, elle est suffisante une fois validée par un humain.

La génération de résumés est une autre tâche dans laquelle les modèles de langage excellent, cette tâche faisant généralement partie de leur entraînement. En travaillant sur le *prompt* et en sélectionnant des exemples probants, la qualité des résumés générés par un LLM est suffisante pour la plupart des applications.

5.3. Applications possibles dans l'INDG

Dans le domaine géospatial, les métadonnées sont très importantes et de qualité variable. Les modèles de langage peuvent être utilisés pour mesurer la qualité des métadonnées présentes dans un catalogue et aider les propriétaires des données à les améliorer.

Concrètement, une application qui utilise un modèle de langage pourrait être développée pour:

- Mesurer la qualité des métadonnées présentes dans geocat.ch, en donnant un score à chaque fiche de métadonnées.
- Assister à la création de fiches de métadonnées en se basant sur des informations existantes tel que :
  - WFS GetCapabilities
  - WFS GetFeature
  - Les jeux de données existants de même type appartenant à d'autres organisations (le jeu de données du cadastre du canton A devrait avoir une description similaire au même jeu de données dans le canton B).
  - Les thésaurus des standards de description de métadonnées
- Traduire automatiquement les fiches de métadonnées dans les différentes langues nationales.
- Générer des synonymes ou des descriptions simplifiées (avec le vocabulaire du public cible) pour faciliter la recherche.

L'objectif serait de fournir une assistance, un copilote, pour la rédaction de fiches de métadonnées et ainsi diminuer le temps passé par les propriétaires de données à remplir le catalogue.

5.4. Exemples et illustrations

5.4.1. Enrichissement et génération de métadonnées

Pour créer ou améliorer des fiches de métadonnées, il est possible de générer une proposition de fiche en se basant sur des informations existantes et sur des modèles de langages. Cette proposition peut ensuite être complétée et corrigée par le propriétaire du jeu de données, puis traduite automatiquement, avant d'être validée et publiée dans un catalogue. L'outil agit comme un assistant, dans un mode *copilote*, qui permet aux gestionnaires de données d'être plus efficaces et de gagner

du temps. Ces derniers restent responsables des métadonnées insérées dans le catalogue. Ils corrigent, améliorent et valident la proposition de fiche de métadonnées.

Les sources utilisées pour construire une proposition de fiche de métadonnées sont celles décrites dans la section précédente (GetCapabilities, GetFeature, layers similaires, thésaurus). A partir d'une fiche de métadonnées préconstruite à l'aide des informations succinctes du WFS, le LLM recherche des métadonnées similaires mieux décrites dans d'autres catalogues, par exemple depuis les données geo.admin.ch ou geocat.ch. A partir de cette base de connaissances (catalogue rempli avec des métadonnées de qualité), la recherche sémantique trouve une série de métadonnées qui servent au LLM pour enrichir la métadonnée. Les métadonnées existantes de qualité peuvent être utilisées par un LLM pour faire une proposition de métadonnées pour un jeux de données similaire.

5.4.2. Enrichissement des métadonnées de geocat.ch

La qualité des métadonnées dans le catalogue geocat.ch est variable. Certaines métadonnées ne contiennent par exemple aucun lien vers la ressource décrite, par exemple un lien web type WMS pour visualiser la donnée géographique.

Souvent, à la place de ce lien, la métadonnée dispose d'un lien vers un site externe, qui lui référence la donnée géographique, par exemple la métadonnée des axes routiers nationaux du canton du Valais qui redirige vers un catalogue ArcGIS.

Dans ce cas, le LLM peut parcourir les sources de ce lien externe pour en extraire les informations relatives à une couche de données type WFS ou ArcGIS REST API, pour la référencer directement dans la métadonnée.

Nous pouvons imaginer bien d'autres façons d'enrichir les métadonnées, les exemples mentionnés ci-dessous permettent d'illustrer les possibilités des LLM dans ce contexte.

6. Conclusion

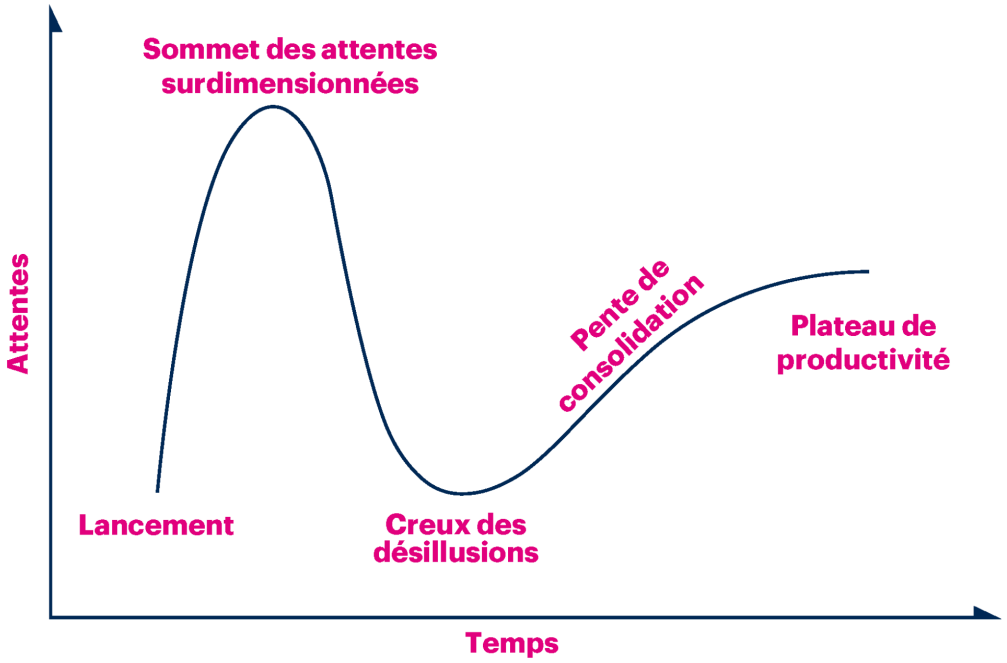
En conclusion de ce document, il nous semble intéressant de comparer l'arrivée de l'IA générative avec l'apparition d'autres technologies disruptives comme l'informatique, internet ou le téléphone mobile.

Ces sauts technologiques majeurs ont en partie changé le monde dans lequel nous vivons. Mais cela a pris un certain temps. Il a fallu des années pour que l'utilisation de l'informatique, d'internet ou du téléphone et de l'internet mobile se généralise.

En ce qui concerne l'IA générative, comme pour d'autres sauts technologiques, **l'impact à court terme est probablement surestimé alors que l'impact à long terme est sous-estimé.**

Il nous semble donc capital de prendre le train de l'IA générative, sans pour autant se précipiter et penser qu'elle doit être présente dans toutes les applications dès l'année prochaine. Les interfaces homme-machine vont être fortement impactées, mais cela ne se fera pas du jour au lendemain.

Une conséquence de la surestimation de l'impact à court terme est qu'il risque d'y avoir, après l'enthousiasme actuel, une phase de désillusion. Cette période pourrait être accompagnée par la faillite de nombreuses startups et l'abandon de projets en relation avec l'IA générative dans beaucoup d'entreprises et administrations en raison d'attentes exagérées. Après l'engouement et l'excitation générée par l'IA générative ces deux dernières années, il n'est pas impossible que 2025 soit l'année de la désillusion. Cela n'enlève rien à l'impact à long terme qu'aura cette technologie.



Gartner hype cycle (image: gartner.fr)

Ce cycle d'attentes exagérées suivit de désillusions avant d'arriver enfin à une utilisation productive de la technologie est souvent illustrée par le "Gartner hype cycle" décrit dans l'image ci-dessus.

Dans le cadre de l'adoption de l'IA Générative dans l'INDG, il faut trouver la voie qui permet de ne pas rater le train tout en évitant de créer des attentes exagérées qui conduirait à des déceptions et donc à l'arrêt des projets dans ce domaine. Les premiers pas doivent permettre de monter en compétence sur les particularités des projets d'IA générative, sans impacter la qualité des services proposés. Ceci passe par la sélection de projets à faible risque et haute valeur ajoutée ainsi que par une communication soignée qui tempèrent les attentes exagérées.
Nous pensons que les domaines comme l'amélioration des métadonnées en mode copilote ou l'introduction de la recherche améliorée à l'aide de modèles de langage sont des candidats intéressants. En parallèle, le domaine nouveau est très dynamique des CUI, qui apportera à terme une véritable révolution au niveau de l'expérience utilisateur, doit être suivi de près. Ceci peut être fait au moyen de veille technologique mais des expériences pratiques, sous forme de PoC, sont nécessaires pour juger de la maturité du domaine et de son impact.