# Mardi 17 septembre 2024

Nous allons commencer la journée par une réunion à travers laquelle nous allons designer l’environnement de test du Copilot. Cette étape est cruciale afin de correctement développer notre Copilot et de surmonter ses défis.

* Premièrement en terme **accessibilité aux fichiers** **et droits**. Il est important de souligné que nous utiliserons le Sharepoint entier de l’entreprise comme database de retrieval pour notre Copilot (puisque la plupart des fichiers des collaborateurs s’y trouvent).   
  Néanmoins, tous les collaborateurs **n’ont pas les mêmes droits d’accès**.   
  Ces droits doivent être conservés à travers l’utilisation du Copilot pour l’ensemble des collaborateurs, pour éviter des fuites de données en interne.
  + Pour ceci nous allons créer un Copilot de test au sein duquel Jérôme et moi allons avoir des droits différents sur des fichiers différents.
* Deuxièmement nous allons effectuer une série de test concernant **le prompt engineering**. Il est à noter que la plupart des **collaborateurs ne sont pas familier avec l’utilisation de l’intelligence artificielle**, il est donc fort probable qu’ils n’aient pas de compétences accrues en prompt engineering. Or, une intelligence artificielle va être plus performante lorsque le prompt d’entrée est correctement formulé (ex : requête lummaps effectuée par Jérôme).
  + Afin de surmonter ce défi, nous avons eu l’idée de créer une boucle d’amélioration du prompt d’entrée, permettant à l’ensemble des collaborateurs d’obtenir de meilleurs résultats de sorties et éviter le bullshit-in/ bullshit-out.
* Troisièmement, nous devons **mesurer les quotas de messages envoyés par les utilisateurs** ce qui apporte un **problème éthique** concernant l’étape précédente : effectivement, nous avions premièrement penser qu’il serait nécessaire que le copilot demande un retour utilisateurs après la boucle d’amélioration pour vérifier si le prompt amélioré est bien représentatif de la requête initialement envoyée par l’utilisateur.   
  Ce qui signifierait qu’au début de chaque conversation avec le copilot, l’utilisateur enverrait 2 messages. Bien que cela ne fasse pas une grande différence sur 1 conversation, Gecina représente 500 collaborateurs, multipliant considérablement le nombre de messages envoyés.
  + Il y a donc 2 solutions possibles qui présentent chacune des problèmes différents :
    - Ne pas demander de feedback utilisateur après la boucle d’amélioration de prompt pour limiter le nombre de messages envoyés. (Solution économique, limitant le quota de message envoyés).
    - Demander un feedback utilisateur, pour s’assurer que le prompt amélioré correspond bien à la demande de l’utilisateur, évitant le risque de fournir un résultat hors-sujet et engendrer des problèmes liés à l’utilisation du copilot (Solution éthique, limitant les problèmes liés à l’utilisation du copilot).
    - Il y a une 3ème option, qui serait d’intégrer une boucle ReAct (Reasonning and acting) de manière à uniquement demander un feedback utilisateur lorsque la différence sémantique entre le prompt d’entrée et le prompt amélioré est faible (et vice-versa). Il faudrait donc définir (peut-être en utilisant du few-shots) une différence sémantique maximale et minimale.

Rappel des demandes opérationnelles :

* + - * Performance / Accuracy
      * Budget
      * Droits
      * Typologie de document (évaluer la performance du copilot en fonction du type de document) (OCR ?)
      * Est-ce que la date d’ouverture ou d’ajout du document dans le sharepoint a-t-il un effet sur la recherche sémantique ?
* Liste des éléments à mettre en place pour l’environnement de test du copilot :

1. Vérifier l’attribution des droits d’accès à certaines informations.
2. Chercher le nombre de documents à partir duquel le copilot perds de l’accuracy pour un prompt de base (ex : lummaps).
   1. Cette information peut-être déjà déterminée et à trouver sur internet où reste à déterminer.
3. Insérer des documents considérés « pointus » de manière à évaluer la capacité du copilot à effectuer de retrieval d’information précise.
   1. Trouver un document considéré difficile > le passé dans ChatGPT qui nous servira d’oracle (modèle et performance de référence) afin de déterminer les critères de succès du retrieval d’information (au sein même du document).
4. Déterminer l’effet de la qualité du prompt sur la qualité des résultats et déterminer s’il est nécessaire de créer la boucle d’amélioration de prompts d’entrée.

La fiche d’évaluation des différents défis évoqués a été créée et est prête à l’emploi.  
  
Jérôme et moi avons discuter de cette approche. Il en a été conclu que cette approche paraissait pointilleuse aux premiers abords, et qu’il n’était pas nécessaire d’appliquer une méthodologie si appliquée pour la conceptualisation du Copilot dans un premier temps. Il était question de time-management et la mention d’une approche plus pragmatique a été soulevée.  
Néanmoins, cette fiche d’évaluation va être conservée pour que nous puissions monitorer le développement du Copilot de manière plus sûre.