Rapport Post-Mortem
Projet BRAIN

FILOCHE Léo
MORLOT-PINTA Louis
DE ZORDO Benjamin
LEGRAND Quentin

1. Introduction

a. Projet

Notre projet avait pour objectif de créer une interface permettant réduire la charge de travail des médecins en proposant une prédiction aussi précise que possible de gliomes à partir d'IRM.

L'interface devait être maniable, simple d'utilisation et respecter la sécurité des données d'un point de vue médical, d'autre part une fonctionnalité de visualisation des IRM devrait être implémentée.

L'IA doit permettre à partir de n'importe quelle IRM (différentes machines, différentes orientations, différents contrastes…) de prédire avec un accuracy minimale de 90% la position du gliome et des ces différentes classes en 3D (IRM).

b. Les équipes

Pour la réalisation de ce projet nous avons été un groupe de 4. Nous nous sommes très vite dispatchés le travail et avons constitué deux groupes de travail, l'un s'occupant de l'interface web (backend et frontend) et l'autre sur l'IA.

Le projet a atteint son objectif final mais aurait pu être amélioré avec un peu plus de temps ou en évitant quelques erreurs que nous avons pu faire lors du projet.

Ce document à pour but de faire une rétrospective de notre projet en listant nos échecs et les solutions trouvées afin d'en tirer des leçons qui pourront nous être utiles pour d'autres projets.

"N'ayez pas peur de faire une erreur. Mais faites en sorte de ne pas faire la même erreur deux fois."
(Akio Morita)

Chacune des parties suivantes sera séparée en trois parties, l'une rédigée par l'équipe interface, l'autre part l'équipe IA et la dernière par l'ensemble du groupe sur les problèmes globaux rencontrés.

2. Facteurs d'échec

Cette partie est une analyse des facteurs qui ont contribué à l'échec, ce sont les petits détails qui nous ont empêché d'avancer correctement ou qui nous ont demandé du travail de recherche supplémentaire.

Interface :

- Choix et installation d'une librairie de visualisation

L'ensemble de l'application se base sur l'utilisation de fichiers d'IRM de cerveau. Or ceux-ci sont encodés au format .nii, et souvent compressés en .nii.gz. Ce sont des fichiers très spécifiques, et il existe donc peu de librairies de visualisation les concernant. Nous avons donc passé une grande partie du début de ce projet à faire un état de l'art et trouver celle qui nous correspondait (n'étant pas du domaine et n'ayant que très peu de connaissances en 3D, elle se devait d'être facile à utiliser, et à installer).

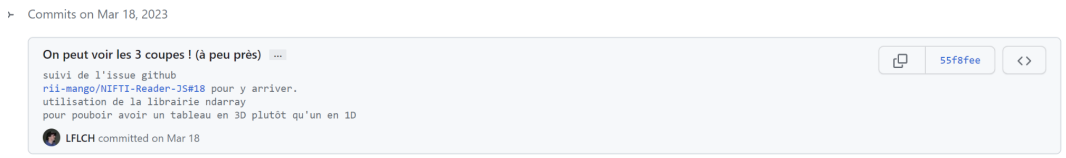
A la fin du semestre dernier, nous sommes finalement arrêtés sur une librairie : Nifti-Reader Js, et c'est celle que nous avons conservée par la suite. Cela nous a contraint à utiliser un autre framework web qu'Angular (le seul que nous connaissions jusqu'alors, et que nous pensions utiliser), car il est basé sur typescript, et que Nifti-Reader Js est seulement en javascript (nous avons essayé de la 'convertir', mais cela s'est avéré plus difficile que prévu et nous avons abandonné cette idée). Nous avons donc appris à utiliser un nouveau framework qui correspondait à nos exigences : Astro.

- Utilisation de la librairie de visualisation et manque de fonctionnalités

Concrètement, l'utilisation de Nifti-Reader Js s'est passée par la modification d'un fichier de démo d'une centaine de lignes de code présent sur le GitHub de la librairie. Cependant, ce fichier de démo était

assez simpliste, permettant l'affichage unique d'un plan (axial). Or nous avions comme exigence de permettre à l'utilisateur de se déplacer en 3 dimensions (et non une seule).

La documentation étant faible, tout comme le nombre d'exemples disponibles, nous avons donc pendant un temps tâtonné, en essayant de gérer ces 3 dimensions nous-mêmes, en vain. Nous étions alors sur le point de changer de librairie, lorsque nous avons trouvé une issue sur le github de la librairie, décrivant exactement notre problème et le résolvant par ailleurs.
Heureux de cette résolution, nous avons fait un commit, indiquant en lien l'issue github associée.



Nous ne le savions pas, mais le fait d'avoir intégré ce lien a fait apparaître le commit dans la timeline de l'issue, rendant visible notre utilisation de la librairie.

C'est alors que le premier avril, nous avons reçu une pull request sur le github du projet, de la part d'un enseignant-chercheur de l'université de Caroline du Sud, contributeur important de la librairie, nous proposant une courte modification du code, avec comme effet d'améliorer grandement l'affichage des images. Nous étions conscients que l'affichage des IRM était dégradé, en comparaison avec leur affichage sur des logiciels adaptés (comme medinria), mais pensions que cela était dû à des limitations techniques de la librairie (et nous envisagions une fois encore de changer de librairie pour cette raison). M Chris Rorden a donc trouvé une solution à un problème que nous pensions irrésoluble !

- Echange de fichiers entre le backend et le frontend

C'est une difficulté à laquelle nous nous attendions à être confrontés, et cela a effectivement été le cas. En effet, l'échange de fichiers via une API REST s'avère plus délicat que l'échange de simples messages au format .JSON. Manquant un peu d'expérience, l'échange de fichiers a pris un peu de temps à être mis en place, mais n'a heureusement pas été bloquant.

- Système de gestion de patients

A l'origine, nous avions imaginé qu'un docteur pouvait accéder aux fichiers de plusieurs de ces patients. C'est une fonctionnalité que nous ne n'avons finalement que partiellement implémentée. Cette notion est présente seulement dans la structure de la gestion de fichiers de la part du backend, mais qui n'apparaît aucunement dans le frontend.
Cela est dû au fait que nous avons repoussé l'ajout de cette fonctionnalité, la jugeant moins importante que d'autres, et que nous avons décidé d'utiliser un système de gestion de fichiers **temporaires** (qui disparaissent à la fin de la session), pour des raisons de sécurité des données.

Intelligence artificielle :

- Notion d'IA

Au niveau de l'IA, beaucoup de notions n'avaient pas été vues au début du projet et étaient pourtant essentielles. Notamment le modèle U-NET qui était un aspect important et qui n'a été vu que tard dans l'année. Il a donc fallu faire des recherches, se forcer à comprendre les projets trouvés sur internet.

- Avancer pas à pas

Même si le sprint 0 nous a permis de voir plus clair concernant l'orientation de notre projet, il était très difficile de prévoir ce qu'il allait se passer en IA pour plusieurs raisons.
La première est celle citée plus haut, n'ayant pas d'expertise dans le domaine il était difficile de prévoir nos avancées et ce qu'il y avait à faire.

La deuxième a été que les projets dont nous nous sommes inspirées n'étaient pas forcément utilisables (librairies propriétaires, matériels spécifiques…) : nous avons passé beaucoup de temps à comprendre ce qui ne marchait pas.

Tous ces facteurs ont fait que nous ne pouvions pas anticiper ce qui allait marcher ou non et ce qu'il restait à faire. Notre objectif premier était de fournir une prédiction, peu importe sa précision.

- Cascade et planification

Par effet cascade des éléments précédents, notre fil directeur était un peu décousu et nos notes de fin de séances n'étaient pas très précises ou inexistantes étant donné que nous n'avançons pas énormément au début (on essayait juste de faire marcher le modèle trouvé).

Global :

- Organisation et charge de travail

Contrairement aux autres groupes, nous n'étions que 4, ce qui c'est ressenti. En effet nous avons décidé d'organiser notre travail en scindant les groupes en assignant à chacun une grande partie : Backend, Frontend et IA.
Même si nous discutions ensemble des problèmes rencontrés, parfois nous restions bloqués ce qui a pu nous décourager par moments.

3. Amélioration et perspective futures

Même si notre objectif final a été atteint, de nombreuses choses auraient pu être faites ou mieux faites avec un peu plus de temps. Cette partie regroupe toutes les améliorations que nous aurions pu apporter.

Interface :

- Système de notifications

Il aurait pû être intéressant de notifier l'utilisateur lorsque certains événements se produisent (perte de connexion avec le serveur, fichiers illisibles, etc…). De par la contrainte de temps, nous avons implémenté l'application en ne prenant en compte que son utilisation 'normale', et en négligeant les cas particuliers. C'est une pratique qui nous a été utile (ne pas s'attarder sur des détails). Cependant, dans l'optique d'un développement de logiciel 'user-friendly', cela pourrait être amélioré.

- Détection automatique du type de fichier

Une fonctionnalité utile pour l'utilisateur aurait été la détection automatique du type de fichier .nii déposé (flair, t1,t1ce ou t2). D'après le standard, cette information est disponible dans leur entête. Pourtant, bien que possédant la clef ('type') dans leur méta données respectives, les fichiers que nous manipulons n'avaient pas de valeur le renseignant. Il faudrait alors chercher d'autres fichiers, ou trouver un moyen détourné si l'on souhaite mettre en place cette fonctionnalité.

- Améliorer le visionneur

Avec plus de temps, nous aurions aimé proposer une fonction de zoom, une fonction d'aperçu en 3D des IRM, ou encore une fonction de calcul automatique du volume de la tumeur. Ce sont des fonctionnalités secondaires par rapport à ce qui était demandé, mais qui s'avéreraient être un vrai plus.

- Meilleur gestion de comptes

Les comptes ont été implémentés de manière sommaire, afin de pouvoir mettre en place un système d'authentification. Cependant, il serait nécessaire de rendre possible la création et la suppression d'utilisateurs via REST, pour respecter la bonne pratique CRUD.

- Déploiement sur serveur

Dès que nous avons réussi à relier le backend et le frontend de sorte à pouvoir faire une prédiction de bout en bout, nous avons voulu déployer la solution sur le serveur de Benjamin, de sorte à avoir une démonstration de livrable fonctionnel dans un contexte de production. C'est une tâche qui n'était pas forcément demandée et qui nous a pris du temps. Cependant, nous avons beaucoup appris en le faisant, et ce sont des notions qui nous seront utiles par la suite.

1) Containerisation de séparation de l'application en microservices

Avant de déployer l'application, il a fallu l'installer sur le serveur. Pour cela, nous avons pris l'initiative de la Dockeriser, afin de la rendre indépendante de l'environnement dans lequel elle évolue.

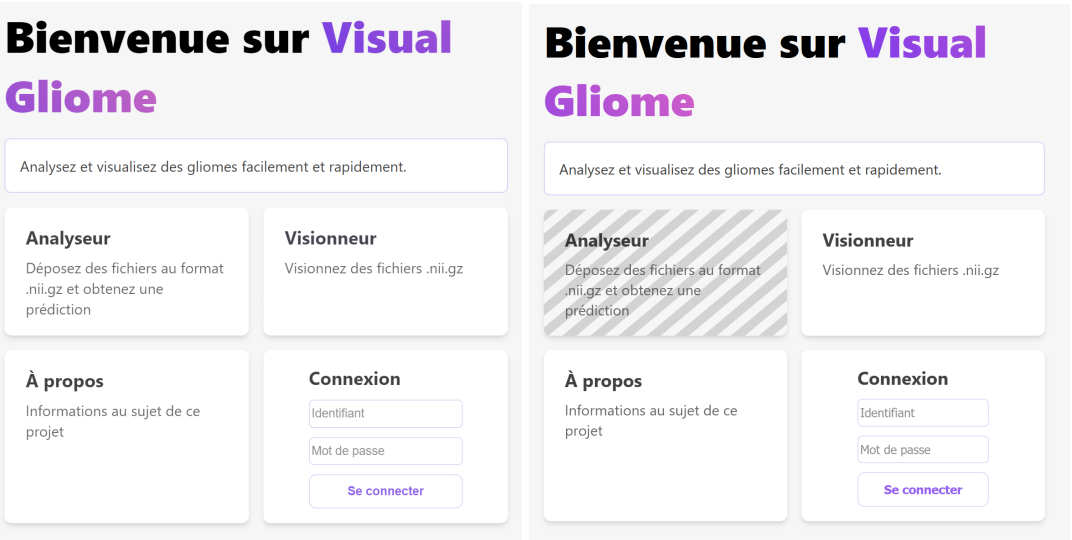
Nous avons dû procéder à un inventaire exhaustif de toutes les dépendances utilisées dans la partie backend de notre application, ce qui a pris un certain temps car il était important de ne pas en oublier.

Étant donné qu'il n'y avait pas de fichier .xml ou .json répertoriant toutes les dépendances du backend (Python), nous avons dû parcourir les imports de chaque fichier et retrouver les bibliothèques associées afin de les indiquer dans un fichier requirements.txt.

Ensuite, il a fallu bien rédiger les Dockerfile, et organiser la communication des containers dans un docker-compose.yml. C'est une tâche qui prend toujours un peu de temps, surtout que cela n'était que la deuxième fois que nous le faisions.

2) Communication backend / frontend lors du déploiement

C'est ce qui nous a posé le plus de problèmes. Malgré que l'application dockerisée fonctionne parfaitement lorsque nous l'exécutions en local, nous avons rencontré beaucoup de difficultés de communication entre le frontend et le backend lorsque celle-ci se retrouvait déposée sur le serveur, rendant impossible la prédiction (car elle nécessite d'être authentifié). Malgré tout, le visionneur fonctionne quoi qu'il arrive car il ne nécessite pas cette fameuse connexion.
Heureusement, après plusieurs jours et plusieurs dizaines d'essais, nous avons (à peu près) réussi à résoudre cette connexion. En effet à peu près car il y a toujours un soucis si l'on se connecte en https, et non en http. Sur google chrome, l'utilisateur est automatiquement redirigé vers le https, ce qui fait que cela ne fonctionne pas. En revanche, cette redirection automatique n'a pas lieu sur Firefox, ce qui fait que cela fonctionne. Allez sur http://visualgliome.bdezordo.com/ pour vérifier par vous même !



Connexion non fonctionnelle (à gauche), et fonctionnelle (à droite)

Intelligence artificielle :

- Le modèle

Avec du recul, les ressources (temps, personnes, connaissances) n'étaient pas suffisantes pour pouvoir créer notre propre modèle (trouver les bons paramètres etc). Néanmoins il aurait été possible d'entraîner le modèle existant sur nos données.

- Meilleur combinaison de contraste

Le modèle utilisé a été entraîné avec deux contrastes et le fichier de segmentation (ground truth). Un aspect d'optimisation aurait été de tester différents contrastes afin de voir quelle combinaison fournissait les meilleurs résultats.

- Prétraitement et post traitement

Durant nos recherches, beaucoup de projets incorporent des prétraitements afin de modifier consciemment l'ensemble d'entraînement (rotation de l'image, réduction du format etc) et pouvoir réduire le surapprentissage du modèle.

D'autre part, de nombreux post-traitement auraient pu être mis en place même si certains traitements ont été nécessaire pour fournir une prédiction du même format que le fichier d'origine.

Global :

- Une application

Le but principal aurait dû être de mettre en place une application pour intégrer notre prototype sur tablette. Cela n'a pas pu se faire puisque les cours de Prog Mobile n'étaient que plus tard dans l'année.

Néanmoins, ayant choisi de faire une interface web, et l'ayant designé pour la rendre responsive, il est (déjà) possible d'y accéder sur un ordinateur, une tablette ou un smartphone depuis un navigateur. La suite logique serait d'intégrer cette application web dans une application (dans une webview Android, ou avec le framework electronjs par exemple)

- Serveur

Certaines choses auraient pu être améliorées, comme dédié un nom de domaine spécifique a Visual Gliome et utiliser un autre serveur que celui personnel d'un des membres de l'équipe. Néanmoins comme ce projet n'est "réel", de telles dépenses n'étaient pas nécessaires.

4. Leçons apprises et bonne pratiques

Après avoir cité les problèmes rencontrés et fait une rétrospective sur l'ensemble de notre projet, vient le temps de lister les leçons apprises. Le but est de dresser une liste de règles à respecter pour nos futurs projets.

Dans l'ensemble notre projet a été plutôt bien mené, cependant les séances de 4 heures sont longues, surtout si l'on travaille sur un même sujet et que l'on ne voit pas les avancées. Nous pensons que **faire des pauses régulières autant individuelles et collectives est important**.

Du point de vue du groupe d'IA, il a été motivant de devoir se plonger dans un projet complexe sans avoir au préalable eu connaissance des notions. **Il aurait fallu étendre le sprint 0 afin de bien comprendre les notions**. Malheureusement nous étions pressés par la deadline de la première présentation.

Toujours au niveau de la partie IA, nous avons passé beaucoup de temps à **essayer de faire marcher des modèles qui n'étaient pas concluants**. Cela nous a **pris énormément de temps** mais nous a également **forcé à comprendre la démarche**. Nous ne savons pas réellement quelles leçons en tirer mais c'était un point important à évoquer.

En ce qui concerne la partie interface, nous avons souvent été freinés par manque d'expérience (recherche de librairies adéquates, compréhension des frameworks et des formats, etc). Heureusement, dans la plupart des cas, ces freins n'ont été que des obstacles à surmonter et non des murs. De plus, nous relatons avoir beaucoup échangé, afin d'avoir un ensemble cohérent de fonctionnalités entre le backend et le frontend. De plus, l'ajout de la partie IA au backend a été un vrai challenge car il a fallu une bonne compréhension de ce qu'il s'y faisait, et de rendre son intégration harmonieuse. En un mot, nous retenons de ce projet **qu'une bonne communication** entre les différentes parties est essentielle, et que cela constitue un élément clé pour que le projet soit mené à bien. En ce qui concerne notre projet, c'est un aspect que nous avons bien pris en compte, et nous n'avons eu ni tensions entre les membres, ni "big-bang" lors des unifications de parties.