Documentation HostoMytho

Bertrand REMY

28 décembre 2024

Résumé

Documentation technique détaillant les logiques d'HostoMytho, les étapes pour déployer l'application en local, se connecter au serveur, et assurer sa maintenance.

1 Objectifs de l'application

HostoMytho est un "jeu ayant un but", développé dans le cadre du projet ANR. Il a pour but de permettre à des utilisateurs d'annoter des compte-rendus médicaux synthétiques (générés automatiquement par différents modèles de langage). Les joueurs qualifient la plausibilité des textes (en évaluant la qualité de la langue et le réalisme médical), identifient et classifient les erreurs (médicales, linguistiques, ou autres), et collectent d'autres données langagières. Les annotations produites par les joueurs sont utilisées pour affiner les modèles de langage grâce à des techniques de fine-tuning.

2 Logique de fonctionnement des jeux

Les jeux affichent des textes ou des passages de textes, dont la longueur est paramétrable. Plusieurs cas de figure sont possibles :

— **Textes générés** : La majorité des textes sont générés, et les annotations à faire dessus ne sont pas connues à l'avance. Les réponses sont donc toujours affichées comme bonnes aux joueurs

— **Textes de contrôle** : Certains textes de contrôle, avec des réponses connues, sont affichés. Ils permettent d'évaluer les joueurs. En cas de mauvaise réponse, le joueur entre dans une boucle de tests et doit fournir une réponse correcte. Cela permet de détecter discrètement les comportements suspects. Les mauvaises réponses font baisser la note de fiabilité des joueurs, les bonnes réponses la font augmenter.

— **Textes déjà traités** : Des textes déjà annotés par d'autres joueurs sont affichés pour renforcer les annotations existantes, et comparer les réponses des joueurs entre eux. Ces textes impactent également la fiabilité des joueurs, mais moins fortement.

2.1 Poids des réponses

Les réponses des utilisateurs ont un poids, déterminé par leur taux de fiabilité (indice sur 100). Ce poids renforce la pertinence des annotations : par exemple, un joueur avec une fiabilité de 90 attribue un poids de 90 à ses réponses, permettant ainsi de calculer des moyennes pondérées. De plus, un joueur médecin (statut déclaré lors de la création du compte) voit ses réponses avoir un supplément de 30% sur le poids correspondant à sa fiabilité.

Exemple de plausibilité moyenne d'un texte, calculée avec les réponses des joueurs dans MythoOuPas :

— Joueur 1 (**J1**) : Fiabilité = 100, Réponse donnée = 75%.

— Joueur 2 (**J2**) : Fiabilité = 50, Réponse donnée = 25%.

— Joueur 3 (**J3**) : Fiabilité = 40, Réponse donnée = 0%.

1. **Somme pondérée :**

$$\text{Somme pondérée} = \sum(\text{plausibilité} \times \text{poids})$$

$$\text{J1} : 75 \times 100 = 7500 \quad \text{J2} : 25 \times 50 = 1250 \quad \text{J3} : 0 \times 40 = 0$$

$$\text{Total} : 7500 + 1250 + 0 = 8750$$

2. **Moyenne pondérée :**

$$\text{Moyenne de fiabilité pondérée} = \frac{\text{Somme pondérée}}{\text{Somme des poids}} = \frac{8750}{100 + 50 + 40} = 46.05$$

La plausibilité moyenne du texte est donc de **46.05%**.

2.2 Calcul des gains de points

Chaque jeu a une base de gain de points paramètrable, par exemple 5 pour MythoNo. Cette base peut-être augmentée si le joueur fais plusieurs sélections sur un texte dans MythoNo ou MythoOu-Pas. Chaque selection ajoute 2 points de base. Cette base de points est ensuite multiplié par l'indice de fiabilité du joueur / 80, puis multiplié par le coefficient multiplicateur du joueur. Ce coefficient correspond à 1 + 0.1 par hauts faits du joueur.

Exemple :
Un joueur ayant une fiabilité de 90 et 6 hauts faits validés réalise deux annotations sur un texte dans MythoNo :

$$\text{Points gagnés} = (\text{Base de points} + 2 \text{ x Annotations}) \times \frac{\text{Fiabilité}}{80} \times (1 + 0.1 \times \text{Hauts faits})$$

$$= (5 + 2x2) \times \frac{90}{80} \times 1.6 = 16$$

Le joueur gagne **16 points**.

2.3 Augmentation du pourcentage de chance d'attraper un criminel

Si le joueur a un taux de fiabilité inférieur à 30, il ne gagne plus de pourcentage (pour éviter les spams et les tricheurs). Plus le joueur attrape de criminels, plus le gain diminue. Le calcul se fait par rapport à un exponentiel et à un taux de décroissance arbitraire de 0.14.

Formule :

$$\text{Gain (\%)} = \text{Base de points} \times e^{-0.14 \times \text{Criminels attrapés}}$$

Exemple : Un joueur ayant déjà attrapé 3 criminels, avec une base de 5 :

$$\text{Gain} = 5 \times e^{-0.14 \times 3} = 5 \times 0.657 = 3.285$$

Le joueur gagne **3.285** points de pourcentage supplémentaires.

3 Explication des données générées

Les textes sont sauvegardés dans la table `texts`. Toutes les phrases de ces textes sont stockées dans la table `sentences`, qui contiennent le `text_id` du texte auquel elles appartiennent, ainsi que la position de ces phrases dans le texte.

Chaque mot (ou token) est sauvegardé dans la table `tokens`, où figurent le `text_id` et le `sentence_id` pour identifier leur origine. Cette table contient également le champ `position`, permettant de localiser le mot dans la phrase à laquelle il appartient.

Ce fonctionnement permet de travailler sur des ensembles de phrases, et non uniquement sur des textes entiers, ces derniers pouvant être longs et fastidieux à traiter pour les joueurs. Les annotations de plausibilité effectuées dans *MythoOuPas* permettent ainsi d'isoler les passages problématiques d'un texte. Cette table contient toutes les annotations de typage des joueurs dans MythoTypo, avec une clé étrangère vers la table précédente, et une vers les différents types possibles.

Les données spécifiques à chaque jeu sont stockées dans des tables distinctes. Voici un visuel et une description de ces tables :

`user_sentence_specification` **- Annotations de négations/absences (MythoNo)**
Cette table contient les annotations de négation et d'absence effectuées par les joueurs dans *MythoNo*. Elle fait référence au texte affiché via le champ `text_id`, et inclut la liste des positions des mots sélectionnés dans le texte (`word_positions`). Elle contient également le poids de l'annotation, calculé en fonction du statut et de la fiabilité du joueur.

`user_text_rating` **- Notations de plausibilité (MythoOuPas)**
Cette table contient toutes les notations de plausibilité des joueurs dans *MythoOuPas*, exprimées sous forme de pourcentages : 0, 25, 50, 75 ou 100. Elle inclut également le poids de chaque réponse, ainsi que la liste des positions des phrases affichées (`sentence_positions`). De plus, la table contient une clé étrangère (`group_id`) faisant référence à la table `group_text_rating`.

`group_text_rating` **- Groupement des notations (MythoOuPas)**
Cette table regroupe plusieurs notes d'utilisateurs concernant un même ensemble de phrases d'un même texte. Elle calcule la moyenne pondérée des pourcentages de plausibilité donnés par les joueurs, en tenant compte du poids de chaque réponse. Le champ `average_plausibility` contient cette moyenne, et le champ `votes_count` comptabilise le nombre total de votes ayant contribué à cette moyenne. Quand on veut charger un texte déjà joué dans MythoOuPas pour comparer les réponses des utilisateurs, on récupère un texte dans cette table.

Idée de développement à ajouter : il serait intéressant de développer sur le serveur, une fonction permettant de vérifier qu'un texte a été traité dans son entièreté et un certain nombre de fois. Cette vérification pourrait s'appuyer sur l'addition des positions de phrases de cette table. Les textes suffisamment traités seraient alors désactivés, afin de prioriser ceux qui ont été moins souvent annotés.

`user_error_details` **- Erreurs détectées (MythoOuPas et MythoTypo)**
Cette table est alimentée progressivement par les données collectées dans *MythoOuPas*. Elle répertorie les erreurs sélectionnées par les joueurs dans ce jeu. Ces données sont ensuite réutilisées dans *MythoTypo*, où les erreurs identifiées dans *MythoOuPas* sont récupérées. Pour afficher une erreur, le texte correspondant est localisé grâce au champ `text_id`, et l'erreur est mise en surbrillance à l'aide du champ `word_positions`.

À savoir : les erreurs de contrôle à trouver dans le jeu sont également stockées dans cette table. Les champs `is_test`, `test_error_type_id` et `reason_for_rate` ne sont remplis que dans le cas d'une erreur de contrôle.

`user_typing_errors` **- Annotations de typage (MythoTypo)**
Cette table contient toutes les annotations de typage réalisées dans *MythoTypo*. Elle inclut :

— `user_error_details_id` : une clé étrangère vers la table `user_error_details`.

— `error_type_id` : une référence aux différents types d'erreurs possibles.

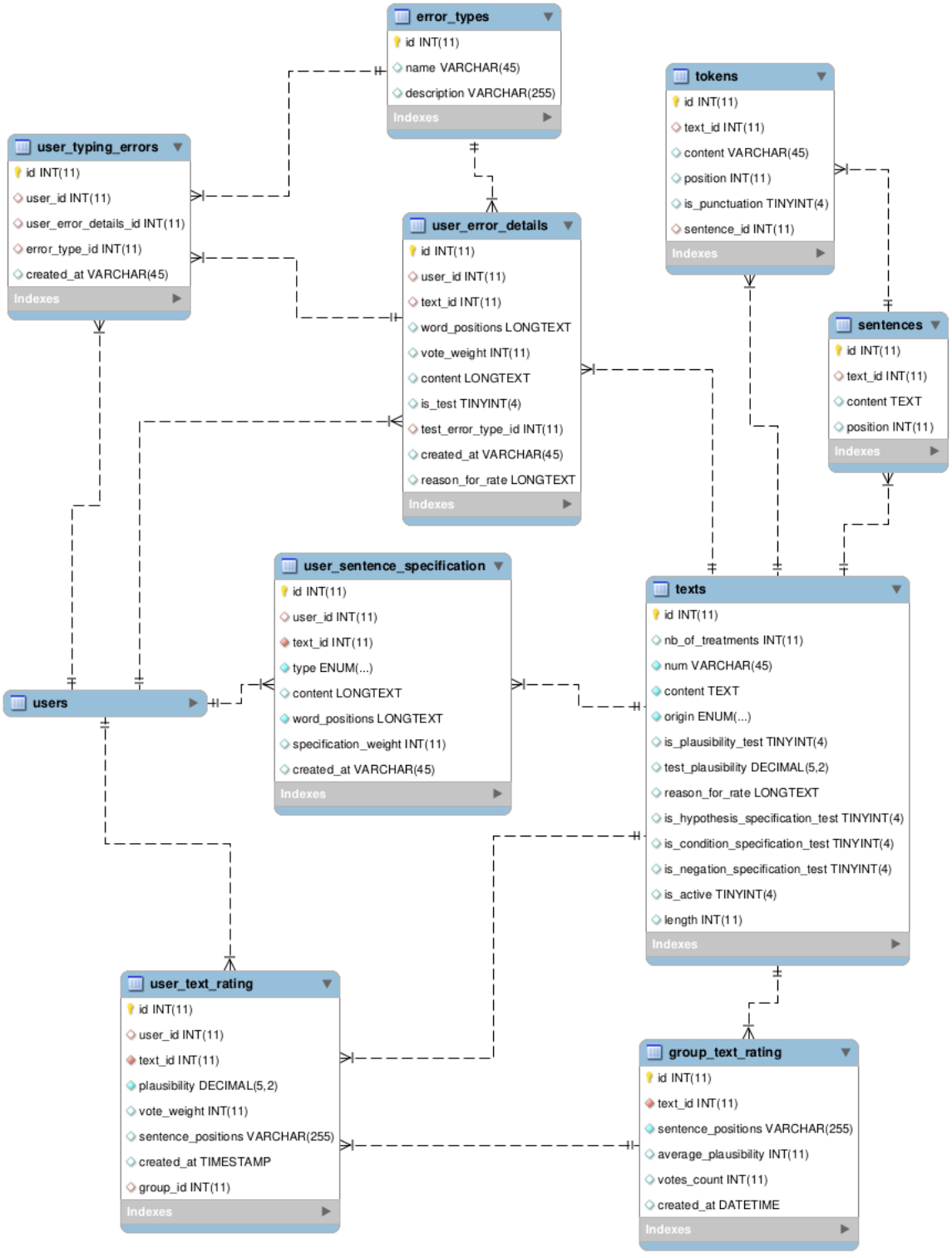

FIGURE 1 – Tables d'annotations

4 Accès à la partie back office

Le back office est accessible via l'url :
https://hostomytho.atilf.fr/HMe1sj

Il faut être connecté avec un compte admin, qui a le champ `moderator` à `true` dans la base de données.

S'il y a des problèmes pour charger les données, essayez de vous déconnecter et de vous reconnecter pour réinitialiser le token de connexion.

4.1 Variables paramétrables

Dans la partie *"Modifier les variables de l'application"*, vous pouvez configurer :

— les gains en points des différents jeux,

— les probabilités d'afficher des textes de contrôle,

— la longueur des textes affichés,

— les gains de pourcentage pour attraper un criminel.

4.2 Suppression d'utilisateur

La suppression d'un utilisateur se fait dans la section *"Gérer les utilisateurs"*. Cliquez sur *"GÉRER"* sur le joueur voulu, puis sur *"Supprimer le compte"*.

À noter que cette suppression ne supprime pas les réponses données dans les jeux par l'utilisateur, ni le compte lui-même, mais anonymise ce dernier en :

— remplaçant le nom d'utilisateur (`username`) par `user_%idUser`,

— supprimant l'adresse email, les points, l'indice de confiance (`trust_index`), la date de création du compte, et toutes autres données permettant d'identifier l'utilisateur.

4.3 Gestion des textes de contrôles dans MythoNo

5 Accès aux sources d'HostoMytho

Source de l'application

https://github.com/HostomythoApp/hostomytho-app

Source du serveur

https://github.com/HostomythoApp/hostomytho-server

6 Mettre HostoMytho en local

Prérequis : installer node, MariaDB, nodemon

6.1 Mettre en place la base de données

1. Installer MariaDB (version du serveur 10.6.18)
2. Pour la structure de la base de données : Créer la base de données et importer la structure de la base avec le fichier `baseBdd.sql` dans les sources du serveur. Elle contient les données nécessaires au fonctionnement de l'app, ainsi qu'un compte modérateur : Pseudo du compte : Azerty mdp : azerty

```
sudo mariadb
CREATE DATABASE hostomytho_bdd;
USE hostomytho_bdd;
SOURCE /chemin/vers/structureBdd.sql
```

3. Pour la base entière avec les données : Se connecter au serveur en SSH (voir la partie connexion SSH), et prendre un fichier SQL backup au choix dans le dossier `server/backups`.

6.2 Lancement du serveur en local

1. Cloner le projet.
2. Renommer le fichier `.envExample` en `.env` et remplir les variables en fonction de l'environnement et de la base de données créée.
3. Exécuter les commandes suivantes :

```
npm install
nodemon start
```

4. Le serveur utilise des scripts python au moment de l'ajout de textes dans le jeu depuis le back office. Pour faire fonctionner ces scripts, il faut donc installer python sur sa machine, créer un environnement appelé hostomythoenv dans le dossier serveur, installer dedans la bibliothèque spacy et télécharger fr_dep_news_trf :

```
 pip install spacy
 python -m spacy download fr_dep_news_trf
```

6.3 Lancement de l'app en local

1. Cloner le projet.
2. Vérifier que dans le fichier `api/index.tsx`, la variable `baseURL` correspond bien à l'adresse du serveur en local (qui s'affiche dans le terminal du serveur à son lancement), ou à l'adresse du serveur en production. Mais toujours effectuer les tests sur le serveur en local.
3. Exécuter les commandes suivantes :

```
npm install
```

```
npm run web
```

ou, pour lancer Expo et avoir plus d'options, comme choisir entre les lancements web/android/ios.

```
npm run start
```

4. Pour relancer une compilation des styles CSS et autres, notamment pour la compilation iOS et Android, exécuter la commande suivante :

```
npm run build:tailwind
```

Le fichier /services/api/index.tsx permet de définir l'adresse du serveur à viser.

7 Connexion SSH au serveur

1. Prérequis : avoir une clé ssh autorisée à accéder au serveur. Demander à Bruno ou Bertrand de donner les droits à votre clé.
2. Pour se connecter au serveur, exécuter la commande suivante :

```
ssh %user%@codeine.atilf.fr
```

Le dossier `www` contient l'application, et le dossier `server` contient le serveur. Les backups quotidiens des 2 dernières semaines sont dans le dossier `server/backups`.

8 Accès à l'App Store

Aller sur : https://appstoreconnect.apple.com
Demander les accès au compte aux concepteurs de l'application.
ou
Créer un compte et demander aux administrateurs l'accès à l'application `hostomytho`.

9 Accès à Google Play pour l'application Android

Aller sur : https://play.google.com/console/u/1/developers

Demander les accès au compte aux concepteurs de l'application.

Pour modifier les informations de l'application (description, images, etc.), suivez ces étapes :
- Allez dans l'onglet *"Store presence"* → *"Store listing"*.
- Cliquez sur *"Edit"* du *"Default store listing"* pour modifier les détails.

10 Mettre en ligne des modifications

10.1 Modification côté serveur

1. Après avoir bien testé les modifications en local, pusher les modifications sur `git` sur la branche `prod`.
2. Se connecter en SSH dans le dossier du serveur.
3. Puller les modifications.
4. Si de nouveaux packages ont été ajoutés, exécuter `npm install`.
5. Relancer le serveur avec la commande suivante :

```
pm2 restart app
```

10.2 Modification côté application web

1. Pusher les modifications sur `git` sur la branche `prod` après les avoir testées en local.
2. Se connecter en SSH dans le dossier `www`.
3. Puller les modifications.
4. Si de nouveaux packages ont été ajoutés, exécuter `npm install`.
5. Compiler l'application avec la commande suivante :

```
npx expo export:web
```

6. Si la compilation s'est bien passée, remplacer l'application web en ligne avec :

```
sudo rm -rf /var/www/web-build
sudo mv web-build/ /var/www/
```

10.3 Mise à jour de l'application Android

Pour compiler l'application, utilisez Expo.
Les builds sont disponibles à l'adresse suivante : https://expo.dev/accounts/hostomytho/projects/hostomytho.
En cas de problème de compilation, essayer différentes versions de npm et de packages. Android et iOS peuvent être sensibles à ça.

Étapes pour générer un APK pour une installation directe sur un téléphone :

– Exécutez la commande suivante :

```
eas build -p android --profile development
```

Étapes pour une compilation d'un .aab destinée à être publiée sur le Play Store :

– Exécutez la commande suivante :

```
eas build -p android --profile production
```

– Une fois la compilation terminée, allez dans la console Google : https://play.google.com/console/u/1/developers/.

– Créez une nouvelle release dans l'onglet *"Production"* en cliquant sur *"Create new release"*.

– Uploadez le fichier `.aab` généré précédemment par Expo.

10.4 Mise à jour de l'application iOS

Comme précédemment, pour compiler l'application, utilisez Expo avec le compte créé précédemment.
Les builds sont disponibles à l'adresse suivante : https://expo.dev/accounts/hostomytho/projects/hostomytho.

Étapes pour une compilation destinée à être publiée sur le Play Store :

– Pour générer l'IPA, exécutez la commande suivante :

```
eas build -p ios --profile production
```

– Une fois la compilation terminée, utilisez un mac, ou une vm mac.

– Téléchargez le fichier IPA et utilisez l'application Transporter pour le transmettre à Apple. Demander les accès au compte aux concepteurs de l'application.

– Une fois transmis, allez sur l'Apple Store Connect : https://appstoreconnect.apple.com/.

– Cliquez sur l'onglet *"Testflight"* et vérifiez que le build a bien été reçu et est valide.

– S'il est bon, allez sur l'onglet `Distribution`, et créez une nouvelle version en cliquant sur le bouton "+" à côté de `App iOS` à gauche.

– Suivez la procédure en sélectionnant bien le dernier build.

11 Scripts automatisés sur le serveur

Plusieurs scripts sont exécutés automatiquement grâce à `cron`. Pour accéder à la configuration :

```
crontab -e
```

Les scripts incluent :

— `month_end.js` : met à jour les gagnants du mois précédent et réinitialise les points mensuels.

— `backup_database.js` : réalise un backup quotidien, stocké dans `/logshostomytho/` pendant 2 semaines.

— `certbot renew` : renouvelle le certificat SSL du domaine.

— `cleanExpiredTokens` : une fonction exécutée depuis `app.js` tous les jours à minuit pour supprimer les tokens expirés.