Xegex

Projets d’annotation par lexique et regex

Encadré par Sarra El Ayari
pour le cours de web, BDD et sites multilingues

Réalisé par Liza Fretel,
étudiante en Master 2 Traitement Automatique des Langues,
2023-2024, Inalco

Table des matières

Préface

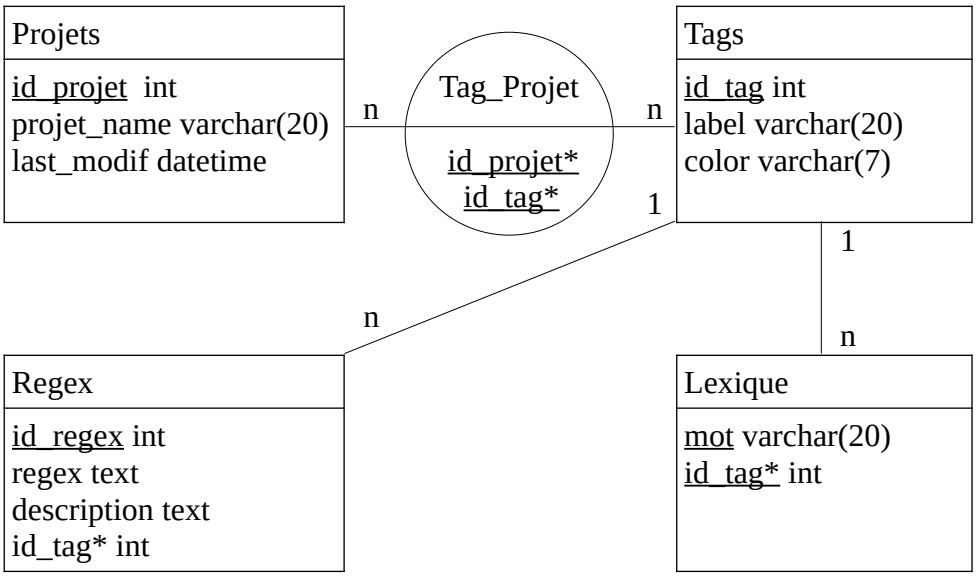
Anciennement étudiante en licence professionnelle DA2I (Développement et Administration Internet et Intranet) à l'Université de Lille, ce projet est pour moi une façon de renouer avec mes compétences en développement web. Je ne l'ai pas pris comme une évaluation, mais comme un projet personnel de développement de compétences. Ainsi, il y a toute une partie utilisant Ajax, qui est une technique que je maîtrise grâce à mes connaissances personnelles, et qui n'est pas 'évaluable' dans le cadre de ce cours. J'ai, cependant, grâce à ce cours et à ce projet, pu développer des compétences nouvelles, notamment en PHP que je n'avais jamais utilisé. HTML, Bootstrap, CSS, jQuery, Javascript, mysql sont tous des langages que je connaissais et dont j'avais un niveau de maîtrise plus ou moins grand avant ce cours, mais grâce au cours j'ai pu revoir les bases sur lesquelles nous étions passés assez rapidement en licence. Ainsi, je remercie Sarra El Ayari d'avoir préparé un cours aussi complet et clair pour le cours de web, BDD et sites multilingues dispensé en Master 2 TAL. Afin d'être évaluée comme il se doit, j'ai tout de même inclus un formulaire 'classique' comme vu en cours, qui transmet ses données à un script PHP qui construit une page HTML.

Description du projet

Xegex est une plateforme web reliée à une base de données SQL dans laquelle il est possible d'enregistrer des projets d'annotation. Ces projets d'annotation consistent en un nom de projet, une liste de tags associés à ce projet, et pour chaque tag, un lexique et des regex qui seront utilisés pour annoter les corpus. Les projets d'annotation peuvent être créés via la page « mes projets ». La page « annotation de corpus » propose, quant à elle, de sélectionner un projet, choisir un ordre de priorité entre les différents tags et envoyer un fichier à traîter. Par envoi d'un formulaire, la page d'« analyse » résume les informations du corpus choisi et du projet d'annotation sélectionné, puis propose un affichage en couleur du corpus annoté et son téléchargement au format HTML ou XML.

Tables SQL

Diagramme UML (un diagramme parle plus que 1000 mots)



Note : id_tag est utilisé dans la table Lexique afin de ne pas avoir deux fois le même mot dans le lexique d'un même tag, et ainsi d'assurer l'unicité des mots du lexique d'un tag. Cela est plus compliqué au niveau des regex, car il faudrait utiliser la variable 'regex' comme clé primaire, or cette variable est de type TEXT et non VARCHAR. SQL interdit de l'utiliser en tant que clé primaire afin d'éviter de faire des comparaisons d'égalité trop coûteuses à chaque ajout d'une nouvelle entrée. En conséquence, il est possible d'avoir, pour le même tag, plusieurs fois la même regex, mais il n'est pas possible d'avoir plusieurs fois la même entrée lexicale.

Initialisation du projet

1. Ouvrez le fichier connexion.php et ajoutez votre nom d'utilisateur et mot de passe mysql. Ce fichier contient les variables de connexion à la base de données utilisé dans le projet.
2. Lancez la commande ./init_db.sh [username] et tapez votre mot de passe mysql deux fois. Cette commande crée la base de données et ajoute quelques valeurs.
3. Lancez la commande ./upload.sh. Cette commande déplace les fichiers dans le dossier var/www/html/xegex en créant au préalable le dossier xegex, puis affiche l'adresse de la page à ouvrir dans le navigateur.
4. Sur la page, cliquez sur « annotation de corpus », sélectionnez le projet POS Tagging, ajoutez le fichier d'exemple JV-Terre_Lune.txt et cliquez

Processus complet CRUD (Ajax et PHP)

Tout le processus CRUD (Create, Read, Update, Delete) est géré par Ajax, sauf l'accès (Read) aux projets, car les projets sont directement affichés lors du chargement des pages (« annotation de corpus » et « mes projets ») et sont donc directement intégrés dans le script PHP projets_manager.php. J'ai fait le choix d'utiliser Ajax afin d'améliorer l'expérience utilisateur et d'accélérer les échanges entre la base de données et la page web. L'interface est, de cette façon, plus intuitive et interactive. Des boutons d'aide sont à disposition (i) à chaque étape (Projets → Tags → Lexique & Regex).

Le code (.mjs et .php) est organisé par Table. Chaque Table possède son script PHP : projets.php, tags.php, regex.php et lexique.php. Lors de l'envoi de la requête vers le serveur, il faut spécifier la méthode POST, mais également le type de requête SQL : 'insert', 'select', 'update' ou 'delete'. Voici la liste des scripts qui sont utilisés pour la gestion des projets d'annotation et de leurs fonctions, avec une description pour chacune d'entre elle. Les scripts mjs et php communiquent entre eux via des requêtes HTTP envoyées avec Ajax :

projets

|__ projets.php	script PHP sans HTML, renvoie des réponses au format json
|__ insert	ajoute un projet dans la BDD
|__ select	renvoie la liste des projets présents dans la BDD
|__ update	met à jour le nom d'un projet si le nouveau nom n'est pas déjà présent
|__ delete	supprime un projet de la BDD

tags

|__ tags.php script PHP sans HTML, renvoie des réponses au format json

- |__ insert ajoute un tag dans la BDD et le lie au projet courant (label et couleur)
- |__ select renvoie la liste de tous les tags d'un projet avec leurs labels et couleurs
- |__ update met à jour le label et la couleur d'un tag si le label n'est pas déjà présent
- |__ delete détache le tag d'un projet (table Tag_Projet)

regex

|__ regex.php script PHP sans HTML, renvoie des réponses au format json

- |__ insert ajoute une regex et sa description dans la BDD en l'associant au tag courant
- |__ select renvoie la liste de toutes les regex pour le tag courant avec leurs descriptions
- |__ update met à jour la regex et sa description, sans se soucier de la duplication de regex
- |__ delete supprime une regex associée au tag courant

lexique

|__ lexique.php script PHP sans HTML, renvoie des réponses au format json

- |__ insert ajoute un mot au lexique du tag courant s'il n'est pas encore présent
- |__ select renvoie la liste de tous les mots associés au tag courant
- |__ update met à jour un mot pour le tag courant s'il n'y a pas de duplication
- |__ delete supprime un mot associé au tag courant

Fichiers javascript qui communiquent avec les scripts PHP

|__ tags.mjs module JS (souligné = fonction exportable)

- |__ createNewTagElements crée les balises HTML du tag. Gère la modification et auto-suppression (Ajax)
- |__ getTagsForProjet récupère les tags du projet sélectionné (Ajax)
- |__ suppression affiche la pop-up de confirmation de suppression (Ajax)
- |__ modification affiche l'input et cache le texte, enregistre le nouveau nom (Ajax)
- |__ ajout clic sur le bouton (+), crée une nouvelle div avec input et enregistre le tag (Ajax)

|__ projets.mjs module JS (souligné = fonction exportable)

- |__ createNewProjectElements crée les balises HTML du projet et les ajoute à page
- |__ suppression affiche la pop-up de confirmation de suppression (Ajax)
- |__ modification affiche l'input et cache le texte, enregistre le nouveau nom (Ajax)
- |__ ajout clic sur le bouton (+), crée une nouvelle div avec input et enregistre le projet (Ajax)

|__ lexique.mjs

- |__ createNewLexiqueElements crée les balises HTML du lexique. Gère la modification et auto-suppression (Ajax)
- |__ getLexiqueForTag récupère le lexique du tag sélectionné (Ajax)
- |__ suppression envoie une requête de suppression du lexique associé au tag courant (Ajax)
- |__ modification clic sur entrée du lexique, affiche l'input et masque le texte, enregistre le nouveau mot (Ajax)

|__ ajout clic sur le bouton (+), affiche le textarea d'ajout du lexique (Ajax)

|__ regex.mjs

|__ createNewRegexElements crée les balises HTML de la regex. Gère la modification et auto-suppression (Ajax)

|__ getRegexForTag récupère les regex du tag sélectionné (Ajax)

|__ suppression envoie une requête de suppression de la regex associée au tag courant (Ajax)

|__ modification clic sur regex, affiche l'input et masque le texte, enregistre la regex et sa description (Ajax)

|__ ajout clic sur le bouton (+), affiche le textarea d'ajout de la regex (Ajax)

Afin de trier les projets par ordre de dernière modification, chaque accès en modification (non SELECT) aux projets ou à un autre objet en ayant ce projet sélectionné dans la page (par exemple, création d'une regex dans un tag d'un projet), modifiera la date de dernier accès au projet.

Notons aussi que nous avons utilisé des modules .mjs car certaines fonctions sont nécessaires entre les différents modules pour assurer le bon fonctionnement de l'application (notamment, le clic sur un tag qui a été créé par tags.mjs déclenche l'apparition des regex et du lexique de ce tag, apparition gérée par des fonctions de regex.mjs et lexique.mjs). Ces fonctions sont soulignées dans le schéma ci-dessus.

1. Création (Create)

Afin de pouvoir ajouter des entrées dans la base de données, j'ai ajouté un bouton (+) pour les quatre tables. Cliquer sur ce bouton fait apparaître des inputs sur la page en fonction des informations dont la table a besoin.

Il y a une certaine difficulté avec la création d'un nouveau tag. En effet, si le nom de tag existe déjà, on n'en crée pas un nouveau, mais on lie l'identifiant du tag existant avec l'identifiant du projet sélectionné (dans la table Projet_Tag). Cela nécessite une vérification préalable de l'existance du label dans la table tags. De plus, si le tag existe, on récupère sa couleur et la renvoie vers la page principale, qui se charge de coloriser directement la balise avec la couleur enregistrée.

2. Accès (Read)

L'accès s'effectue lorsque l'utilisateur sélectionne un projet ou un tag. Nous affichons alors les informations liées à ce tag, en vidant d'abord les balises prévues à cet effet.

Les fonctions select font souvent le lien entre deux tables : par exemple, récupérer les tags qui sont associés à un projet, ou récupérer le lexique qui est associé à un tag. Les projets font exception, puisqu'ils sont tous chargés en même temps avec le chargement de la page.

3. Mise à jour (Update)

Pour la mise à jour, l'utilisateur doit double-cliquer sur le bouton de projet ou de tag ou cliquer sur une regex ou un mot du lexique. Cela fera disparaître la balise de contenu pour la remplacer par une input. Si la valeur de l'input est changée et que l'utilisateur appuie sur ENTREE, la requête de mise à jour est envoyée au serveur.

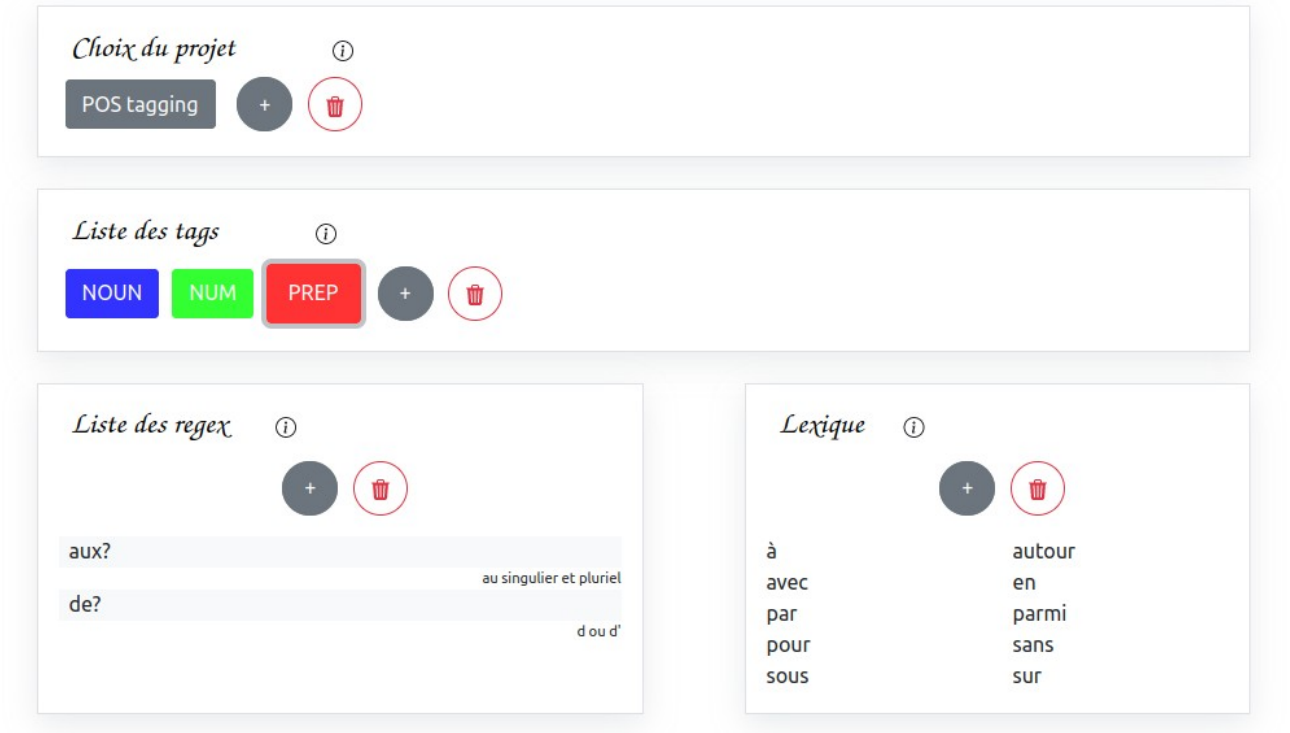
Si un objet est modifié par mise à jour, il le sera dans tous les autres projets d'annotation.

4. Suppression (Delete)

Il est possible de supprimer des entités de la même façon que pour la mise à jour, mais en supprimant tout le contenu de l'input puis en validant. S'il s'agit d'un projet ou d'un tag, l'ancienne valeur de l'entité est affichée dans le message de suppression pour demander confirmation.

La suppression d'un tag n'engendre pas sa disparition totale, mais sa désallocation au projet courant. Autrement dit, un tag ne peut pas être totalement supprimé. Afin de faciliter la liaison d'un projet avec un tag existant, on utilise un menu déroulant affichant des suggestions de tags, qui est le résultat d'un select de tous les tags pas encore liés au projet.

Voici une capture d'écran de la page de gestion de projets :



Annotation de corpus (via envoi de formulaire)

Cette fonctionnalité implémente PHP de la même façon que vu en cours, sans Ajax.

La première page, « annotation de corpus » propose un formulaire de sélection de projet, de désactivation des tags et d'arrangement de l'ordre des tags, d'upload de fichier textuel. De la même manière que la page « mes projets », cette page pré-charge la liste des projets côté serveur en PHP. L'arrangement de l'ordre des tags se fait par glisser-déposer, ce qui va changer l'ordre d'apparition des cases à cocher dans le formulaire et donc leur indice dans le tableau récupéré en PHP. Après envoi, le formulaire est réceptionné par la page « résultats de l'analyse ».

La deuxième page « résultats de l'analyse », dans un premier temps, résume les informations sur le corpus soumis : le nom du fichier, sa taille, ainsi que le projet d'annotation sélectionné et les tags actifs, triés par ordre de préférence.

Dans un deuxième temps, le script PHP effectue l'annotation du contenu du corpus en ajoutant

des balises colorées avec la couleur du tag autour des mots annotés. Le label du tag est inclus dans l'attribut 'name' de la balise . Pour annoter le corpus textuel, nous effectuons une boucle foreach sur chaque <tag> transmis par le formulaire, puis une autre boucle foreach sur tout le lexique de ce tag (récupéré dynamiquement dans la base de données avant le foreach). Nous nous servons d'expressions régulières pour annoter les mots contenus dans le lexique (par remplacement dans la chaîne de caractère du texte et en ajoutant des frontières de mot (\b) autour du mot dans l'expression régulière). Puis, nous appliquons chaque regex associée au tag, une par une, à l'aide d'une autre boucle foreach (les regex sont aussi récupérées dans la base de données avant le foreach).

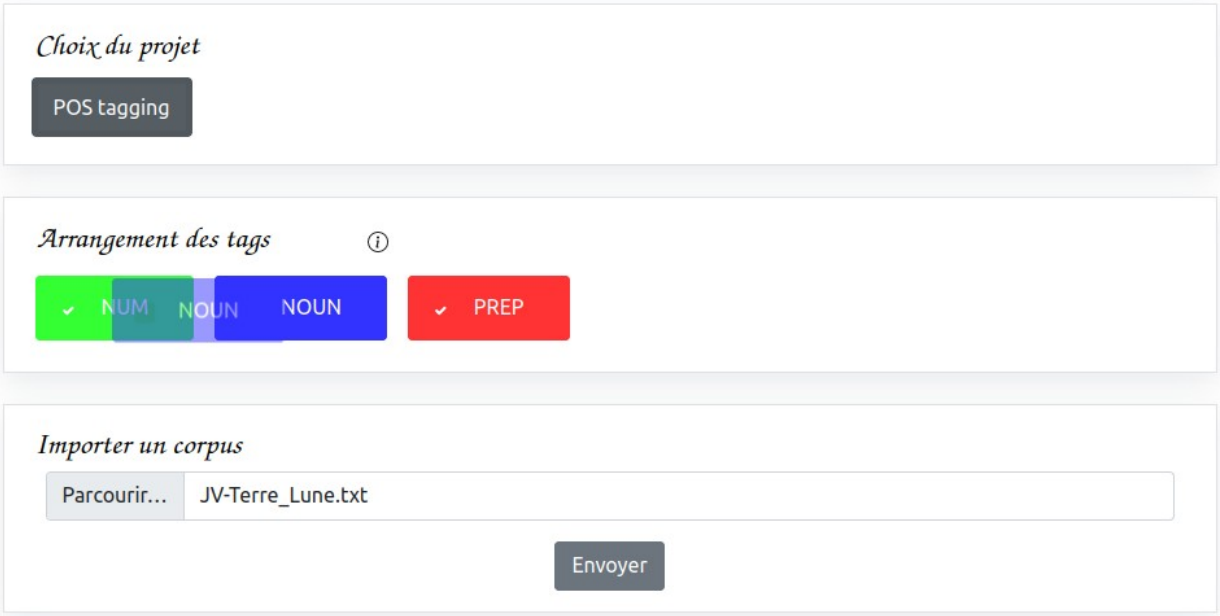
Remarque n°1 : afin du mieux visualiser le label des tags résultants autrement que par la couleur, une fonction javascript affiche une infobulle avec le nom du tag lors du passage de la souris sur la balise.

Remarque n°2 : lors de l'annotation automatique, un problème est survenu pour le traitement d'un corpus en français. En effet, les regex en php ne considèrent pas par défaut les caractères diacrités comme des lettres. Cela est embêtant dans le cas du français : les caractères diacrités sont considérés comme des bords de mots (\b). Il faut ajouter 'u' dans la regex php afin d'indiquer que la regex est en unicode.

Remarque n°3 : comme l'expression régulière effectue un remplacement dans la chaîne de caractère, il est possible qu'elle 'matche' des éléments dans les balises nouvellement ajoutées. Il faut réduire au maximum ce résultat indésirable, soit en enregistrant des indices de début et de fin de chaîne pour chaque tag, soit en diminuant le nombre d'opérations grâce à la factorisation d'expressions régulières. Par exemple, pour toutes les entrées du lexique, les inclure dans une seule grande expression régulière de cette façon : \bmot1\b|\bmot2\b|\bmot3\b non seulement évitera quelques bugs, mais aussi réduira drastiquement le temps de traitement du fichier, avec une seule expression régulière à appliquer sur le corpus pour toutes les entrées du lexique au lieu d'avoir une expression régulière par entrée du lexique.

Dans un troisième temps, la page propose le téléchargement du corpus annoté au format HTML et au format XML. Le traitement est effectué en javascript, lors du clic sur l'un des deux boutons. L'avantage du format HTML est qu'il permet d'afficher les couleurs des balises, il est donc plus pratique pour la visualisation des résultats. Quant au format XML, il est plus pratique pour effectuer des traitements informatiques. Dans le format XML, le nom de la balise est le label du tag directement : <label>. Ce n'est pas un attribut comme dans le format HTML : . La difficulté de la conversion du contenu HTML en XML a été de ne pas modifier l'apparence de la page (autrement dit, les balises présentes dans le DOM), car il a fallu transfomer les balises en balises <label> et supprimer la couleur du tag. Cependant, il n'existe pas de fonction clone() pour copier l'objet jQuery. J'ai résolu ce problème en créant une nouvelle variable qui copie non pas l'objet jQuery, mais le HTML contenu dans cet objet, qui est natif (il ne pointe pas vers un objet et sera donc copié).

Voici une capture d’écran du formulaire :



La liste des projets est récupérée en PHP au premier chargement de la page.

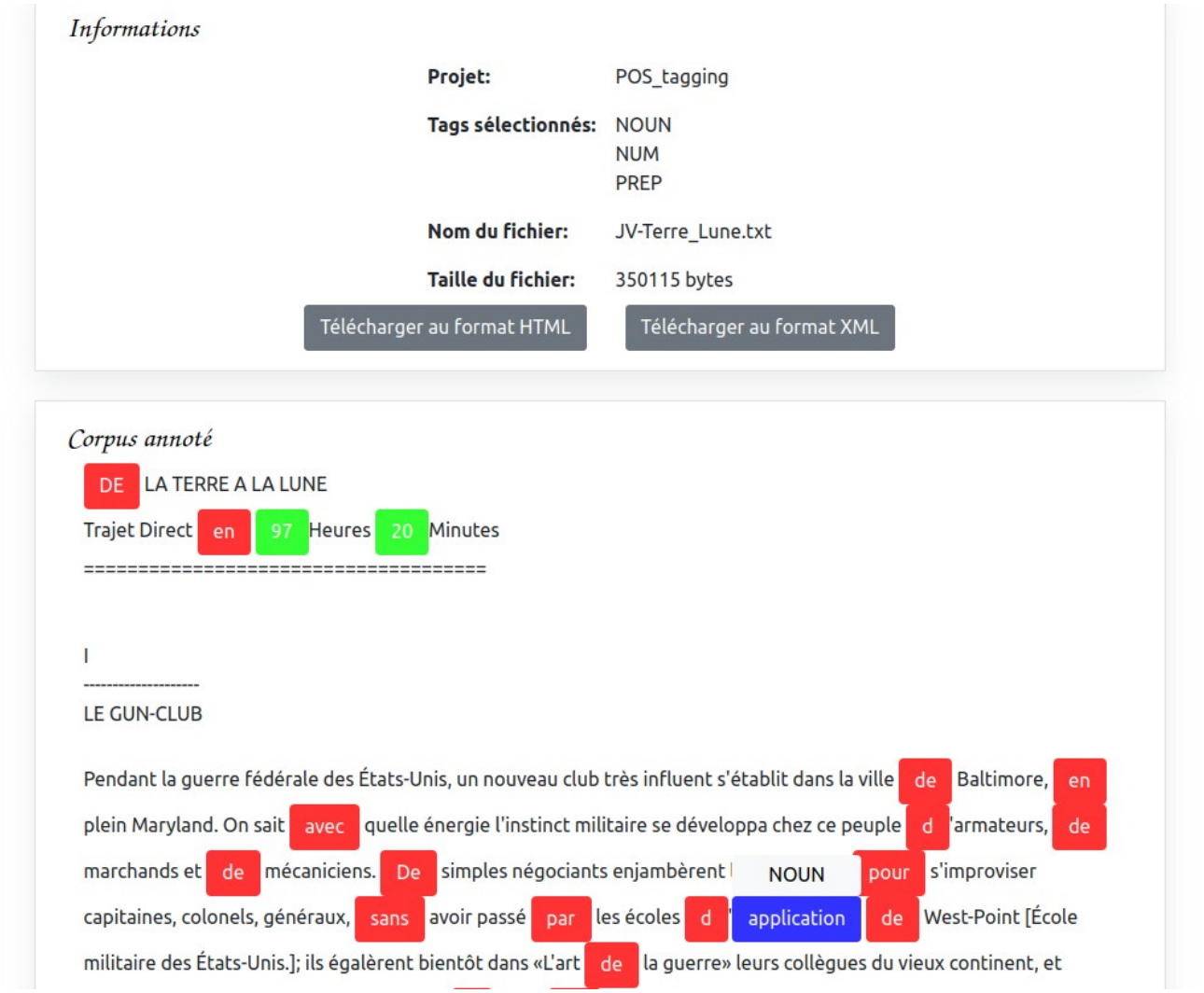
La liste des Tags est récupérée après le clic sur un projet, via Ajax.

L’ordre des tags est interchangeable par glisser-déposer.

Il est possible de désactiver un Tag pour cette analyse uniquement (la désactivation ne sera pas enregistrée dans la BDD, mais sera transmise par l’envoi du formulaire).

Il n’est possible que d’envoyer des fichiers dont l’extension est .txt.

Voici une capture d'écran de la page de résultat de l'annotation :



Voici le résultat attendu au format xml après téléchargement pour le corpus JV-Terre_Lune.txt :

<?xml version='1.0' encoding='utf-8'?>
<corpus><prep>DE</prep> LA TERRE A LA LUNE
Trajet Direct <prep>en</prep> 97 Heures 20 Minutes
==========

I

LE GUN-CLUB

Pendant la guerre fédérale des États-Unis, un nouveau club très influent s'établit dans la ville <prep>de</prep> Baltimore, <prep>en</prep> plein Maryland. On sait <prep>avec</prep> quelle énergie l'instinct militaire se développa chez ce peuple <prep>d</prep>'armateurs, <prep>de</prep> marchands et <prep>de</prep> mécaniciens. <prep>De</prep> simples négociants enjambèrent leur comptoir <prep>pour</prep> s'improviser capitaines, colonels, généraux, <prep>sans</prep> avoir passé <prep>par</prep> les écoles <prep>d</prep>'<noun>application</noun> <prep>de</prep> West-Point [École militaire des États-Unis.]; ils égalèrent bientôt dans «L'art <prep>de</prep> la guerre» leurs collègues du vieux continent, et comme eux ils remportèrent des victoires <prep>à</prep> force <prep>de</prep> prodiguer les boulets, les millions et les hommes.
[…] </corpus>

Fonctionnalités manquantes

Il faudrait pouvoir ajouter, dans les regex et le lexique, la possibilité d'ignorer la casse (i). Cela pourrait être une valeur booléenne et une case à cocher sur la page de gestion des projets au moment de la création ou de la modification du lexique et de la regex. Ensuite, dans analyse.php, on doit vérifier la valeur de ce booléen et enlever ou laisser le 'i' dans les expressions régulières :

```
$regex = "/\b".$mot."\b/iu";
```

$mot : mot du lexique ou regex ; \b : frontière de mot ; i : ignore case ; u : unicode

La possibilité de se connecter aurait été pertinente afin d'avoir plusieurs utilisateurs simultanés qui ne peuvent agir que sur leurs propres projets ou sur les tags qu'ils ont créés. Un utilisateur différent peut créer un projet ou un tag portant le même nom que celui d'un autre utilisateur.

De plus, si l'on veut définir différemment nos Tags mais avec le même nom, par exemple pour deux langues différentes ou des critères différents, cela est impossible car les Tags ont un nom unique.

Pour les langues orientales sans espace telles que le chinois ou le japonais, les corpus doivent être préalablement découpés en raison du \b qui est ajouté par analyse.php. Il aurait fallu laisser à l'utilisateur le choix de ne pas tenir compte des bords de mot (\b) dans les regex et dans le lexique.

Intégration de Bootstrap et du css

Le style choisi pour l'application est un style neutre, épuré. Les couleurs blanches à grises permettent de mettre en valeur les couleurs des tags choisies par l'utilisateur. La plupart des balises ont des classes bootstrap afin de rectifier leur margin et le padding par rapport aux autres éléments et améliorer l'esthétique globale du site.

Certaines fonctionnalités sont plus simples à gérer avec un css qu'avec des classes Bootstrap. Le fichier style.css, importé dans toutes les pages du site, permet notamment de :

- faire disparaître les fieldset pour une apparition progressive (classe 'active') au fur et à mesure que l'utilisateur avance dans les formulaires ;

- pour modifier le comportement d'un type de balise entier (legend, td, li), il est plus simple d'écrire trois lignes dans le css que d'ajouter une classe bootstrap à chacune de ces balises ;

- superposition de logos : par exemple, pour l'envoi d'un lexique, le logo est positionné au-dessus du textarea. J'ai créé une classe .superposable et .objet-superpose ;

- div de confirmation de dialogue : se comporte différemment des autres div, car elle n'est pas visible et lorsqu'elle apparaît, elle se place au premier plan sur la page. Ainsi, ces div ont une classe spécialement conçue : .confirmation-dialog.