GeoDiff

Projet RICM4 - Polytech' Grenoble
2016-2017

BECHER Hervé - BROCHIER Aymeric
AMAURIN Alexandre

Un diff pour les données géographiques

Sommaire

I. Présentation du sujet

L'information géographique peut être géocodée dans de multiple formats : GeoJSON, XML, PostGIS... L'objectif est ici de permettre de construire le delta entre deux versions d'une même information, puis de visualiser ce delta.

Par exemple, deux fichiers GeoJSON représentent la même information mais a des instants différents. Il est donc nécessaire de faire correspondre les éléments qui sont préservés entre deux versions, tout en mettant également en évidence les éléments ajoutés et supprimés.

Il est pour cela possible de s'inspirer des différences sur du code où le code supprimé est en rouge, et le code ajouté en vert. Des versions plus récentes permettent également de détecter les modifications qui sont alors illustrées avec une autre couleur, jaune pour Gumtree par exemple.

II. Interface web

L'interface web est réalisée en JavaScript pour faciliter la manipulation des données GeoJSON. Chaque feature possède un champ Id qui l'identifie de manière unique.

Tout d'abord, l'interface web se compose d'une partie permettant à l'utilisateur de rentrer dans un champs texte le nom de son champ id. Cet identifiant est situé dans les properties de chaque feature. Enfin, l'utilisateur clique sur "ok" pour valider son choix.

Ensuite, l'utilisateur choisit les deux fichiers en entrées. Au préalable, l'unicité de chaque id est vérifiée. Pour ce faire, une table contenant un tableau associatif id-index est utilisé pour chaque "feature" de chaque fichier : si la valeur associée à un id est "undefined" c'est que c'est un nouvel objet et il est ajouté dans la table associée à ce fichier, sinon les id ne sont pas uniques et cela ne permet pas d'aller plus loin. Enfin, le tableau des suppressions est initialisé par toutes les features contenus dans le premier fichier d'entrée et est réduit au cours du calcul.

Après avoir rempli la table des id pour les deux fichiers d'entrée, les features du deuxième fichier sont parcourus un par un : la présence de chaque id du second fichier est vérifiée dans le premier, et si tel est le cas alors ce feature n'est pas une suppression et il est supprimé de la liste des suppressions.

Dans le fichier résultat, une propriété "geodiff-type" a été ajoutée pour marquer les évolutions. Cette propriété prend les valeurs add, del ou mod selon le type de la modification.

Pour le calcul des différences, il s'agit d'un objet global contenant les fichiers d'entrée et 3 tableaux supplémentaires : un pour chaque type de modification.

Ensuite, les features sont comparés par la fonction CompareGeometryGeojson. Elle vérifie l'égalité de deux objets coordinates Geojson et renvoie vrai ou faux. Elle fonctionne pour tous les types d'objets Geojson : Point, Polygone …

Si l'égalité est respectée, c'est-à-dire que nos features sont identiques d'après la fonction de comparaison, alors on ne fait rien de particulier. Si la fonction renvoie faux, alors c'est une modification et une copie de ce feature est ajoutée dans le tableau des modification et on lui ajoute un marqueur de modification. Enfin, si l'id n'existe pas c'est alors un ajout, et on le place dans le tableau des ajouts avec la propriété "geodiff-type" add.

Une fois la totalité du deuxième fichier traité, ces données sont transmises à une page php pour créer une archive contenant différents fichiers correspondant aux ajouts, suppressions et modifications et un fichier contenant la totalité. La méthode POST est ainsi utilisée pour cela et les différentes données sont identifiées par leur type de modification et redirigées dans des fichiers différents.

Finalement, l'utilisateur a la possibilité de télécharger cette archive.

III. Outil en ligne de commande

Cet outil est une version standalone de l'interface graphique, utilisable directement en ligne de commande. Celui-ci prend comme paramètres deux fichiers au format GeoJSON (plus d'autres options détaillées par la suite) et retourne un fichier également au format GeoJSON.

Le format des fichiers en entrée doit être du type FeatureCollection.

(voir la RFC 7946 : https://tools.ietf.org/html/rfc7946 pour plus d'informations sur les types)

1. Utilisation

Cet outil est écrit en Java, sous Java 8, et nécessite donc d'avoir un JRE installé sur le système. Le programme attend deux fichiers en entrée, et plusieurs options peuvent être spécifiées :

- `--metadata` ou `-m`

  Exemple : `--metadata id=MyUid;`

  Les métadonnées apportent des informations aux parsers lors de la construction des objets internes représentant les Features. A l'heure actuelle, ces métadonnées servent à spécifier l'identifiant unique de chaque Feature. En effet, le programme a besoin d'identifier de façon unique chaque élément afin de trouver son équivalent dans les deux jeux de données.

  Le système est extensible et permettra de spécifier plus de paramètres, notamment si des plugins viennent à être développés.

  La syntaxe générale des métadonnées est de la forme suivante :

`--metadata <key>=[value];<key>=[value];...`

Note : le dernier ";" est optionnel.

Si une clé ou une valeur contient un des caractères de délimitation, ceux-ci peuvent être échappés avec le caractère "\".

- `--filters` ou `-f`

  Exemple : `--filters add,del,new`

  Définit les données à conserver en sortie. Plusieurs valeurs sont acceptées, séparées par des virgules : `add` (ajouts), `del` (suppressions), `old` (anciennes versions), `new` (nouvelles versions), `mod` (les quatre précédents réunis), `id` (inchangés) et `undef` (tout ce qui n'a pas été reconnu).

- `--output` ou `-o`

  Exemple : `--output result.geojson`

  Définit le fichier de sortie du programme. S'il n'est pas défini, la sortie se fera dans la console.

- `--data` ou `-d`

  Exemple : `--data geojson`

  Spécifie le format des données d'entrée. Deux formats sont supportés : GeoJSON (`geojson`) et CSV (`csv`) (type Point uniquement).

Pour obtenir de l'aide / la liste des options disponibles du programmes, utilisez l'argument `--help` ou `-h`.

2. Documentation

La Javadoc pour l'outil est disponible sur la page GitHub dans le répertoire javadoc (sous formes brute et compressée) et donne les informations essentielles sur les classes utilisées. Elle rappelle également les cas d'utilisations du programme et donne des informations supplémentaires sur certains types de données.

3. Tehnologies utilisées

- Gson (de Google) pour manipuler des objets JSON
- GeoGSON (addon pour Gson fait par l'utilisateur GitHub filosganga) pour manipuler des données au format GeoJSON
- JOpt Simple pour gérer les options

4. License

La license est encore à décider, les librairies utilisées étant sous les licenses Apache 2.0 (Gson et GeoGSON) et MIT (JOpt Simple), deux licences compatibles. Il est bon de noter qu'aucune modification n'a été apportée aux sources de ces projets.

5. Limites

Cet outil ne prend en charge que les fichiers au format GeoJSON et offre un support minimal au CSV (manipulation de points).

Bien que non requis dans le standard GeoJSON, un identifiant unique pour chaque élément est indispensable au bon fonctionnement du programme. En effet, sans moyen d'identification, on ne pourrait pas détecter les différences.

De plus, le parser utilisé ne tolère pas les géométries vides et ne les ignore donc pas, ce qui lève une exception et interrompt le programme.

IV.Conclusion

En conclusion, nous avons atteint nos objectifs en temps et en heure : l'outil calcule bien le delta entre deux fichiers et ce dernier est bien visualisable grâce à l'interface graphique.

Néanmoins, ce projet a connu un départ difficile du fait de l'absence de compétences en Javascript. Ce projet pourrait encore être amélioré en incluant de nouveaux formats de données : XML, PostGIS… voire en ajoutant une API pour inclure ses propres formats (parsers, éléments, etc.).