Ecole nationale de la statistique
et de l'analyse de l'information

Projet "Wikipedia Matrix"

Etudiant 3A - Master EDP filière STD

Etudiant :
Ulysse Boucherie

Enseignant :
Mathieu Acher

Table des matières

Table des figures

1 Introduction

Wikipedia est une fantastique source de données, principalement composée d'articles écrits en langage naturel (e.g., français, anglais). L'objectif de ce projet est d'extraire des tableaux au format CSV à partir de pages Wikipedia.

1.1 Motivation

Les tableaux Wikipedia sont difficiles à exploiter par des outils statistiques, de visualisation ou n'importe quel outil capable d'exploiter les tableaux (e.g., Excel, OpenOffice, RStudio, Jupyter). Ils sont écrits dans une syntaxe (Wikitext) difficile à analyser et non nécessairement conçue pour la spécification de tableaux. De plus, il y a une forte hétérogénéité dans la manière d'écrire des tableaux, ce qui complique encore plus le traitement des données tabulaires de Wikipedia. Le même constat peut être fait pour le format HTML qui peut être utilisé pour présenter un tableau dans un navigateur Web : il n'est pas facilement exploitable par des outils statistiques ou des tableurs.
L'objectif est donc d'extraire les tableaux Wikipedia et de les traduire dans un format plus simple et adapté ; nous choisissons le format CSV (*comma Separated Value*).

2 Méthode applicative de récupération des tableaux

2.1 Principe général

Comme l'illustre la figure 1, pour récupérer un tableau d'une page Wikipedia, on décide de passer par son format HTML. Grâce à notre `WikipediaHTMLExtractor`, qui appelle notre *Parser* `parseComplexTable(Element htmltable)` qui lui-même récupère les éléments `<table>`, on obtient les tableaux de la page au format CSV. On fait cela pour tous les urls contenus dans notre fichier text et le tour est joué.

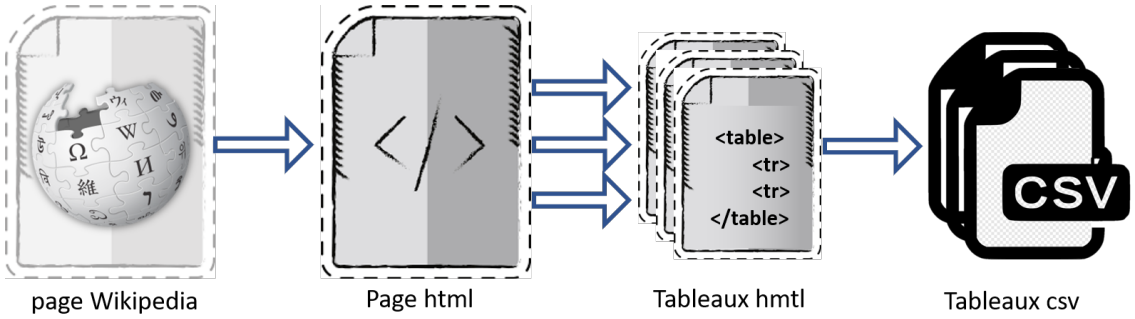

FIGURE 1 – Schéma de fonctionement de l'application

2.2 Classes et *Design Pattern* utilisées

Nous avons créer la classe `Table.java` pour définir un tableau. Les classes `ParseWikitable` et `WikipediaHTMLExtractor` sont des pattern *Adapter* qui nous permettent de rendre compatibles les différents formats de tables mais aussi de fichier !

Nous avons aussi utilisé les patterns *Factory*, *Visitor* et *Chain of Responsibility*, qui nous ont permit d'importer les tables malgré leurs nouveaux formats au fil de l'eau.
Au finale, 11 types de tables wikipedia ont été retenu et 71 rejetés. (voir détail à 5.2)

3 Défis

3.1 Le formatage des tableaux dans Wikipedia

Wikipedia contient énormément de données (du texte, des figures, des sections, etc.) qui ne sont pas des données tabulaires. Il est même possible qu'une page Wikipedia ne contienne tout simplement pas de données tabulaires. Un premier défi a donc été d'occulter ce genre d'informations et ne considérer que les données pertinentes pour l'extraction de tableau. Il a fallu aussi être en mesure d'extraire plusieurs tableaux sur une même page Wikipedia.

3.2 Les tableaux imbriqués

Une autre difficulté a été que certaines données tabulaires étaient difficiles à convertir au format CSV car elles étaient imbriquées. Une première façon était via des *ColSpan* ou *RowSpan* qui signifient que certaines cellules recouvraient plusieurs colonnes ou plusieurs lignes. La deuxième façon était d'avoir une `<table>` dans un autre `<table>`.

3.3 Données hétérogènes

Une barrière supplémentaire concernait l'hétérogénéité des données tabulaires qui a compliqué la tâche d'extraction. Propre à wikipedia, certaines données renvoient à des sources ou des liens internet. Certaines autres faisaient apparaître des symbols qui n'apartiennent pas à l'alphabet latin que l'on connaît bien. Entre autres, des formules de maths ou des sigles d'autres alphabets.

4 Solutions envisagées

Cette partie réfère aux défis abordés précédemment.

4.1 Formatage maté

Pour traiter le plus de tableaux possible malgré le formatage varié des tableaux des pages Wikipedia, nous avons d'abord sélectionné un seul URL pour lequel nous avons compté le nombre de tableaux "pertinents". En regardant la page HTML de cet url nous avons détecté les formats de `<table>` différents de la page et avons créé 2 listes les catégorisant : ceux qui nous intéressaient (`relevantType`) et ceux qu'on ne voulait surtout pas traiter (`ignoreType`). Au fur et à mesure que notre programme trouvait les tableaux des URLs du fichier text, nous obtenions davantages de formats. Tous ceux qui se trouvaient dans les URLs ont été catégorisés dans l'une des 2 listes.

4.2 Tableaux imbriqués traités

Cette méthode n'était pas demandé mais elle nous a permit d'obtenir beaucoup de tableaux. Parfois, l'affichage est lourd et peu informatif. Cependant, il y a de nombreux tableaux qui ne comptaient que quelques `Rowspan` ou `Colspan`. Ainsi, nous avons pris l'initiative de les compter dans nos tableaux pertinents
Une solution qui a été mise en oeuvre pour le premier problème a été de coder un remplissage des cellules comme le montre la figure 2. Plus précisément, la fonction `set(int rawIndex, int colIndex, String value)` de la classe `Table.java`.

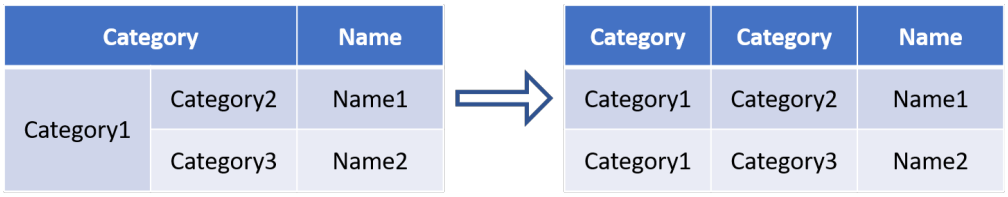

FIGURE 2 – Exemple de remplissage de cellules avec *Rowspan* et *Colspan*

Le problème des tables contenues dans les autres tables a demandé à chercher la doc de Jsoup et de comprendre la notion de node. Mais cela a permis de récupérer les tables contenues dans d'autres tables et d'y ajouter leur contenues, comme l'illustre la figure 3.
Il a été remarqué cependant que la pluspart de ces "sous-tables" ne contenaient le texte que d'une seule case.

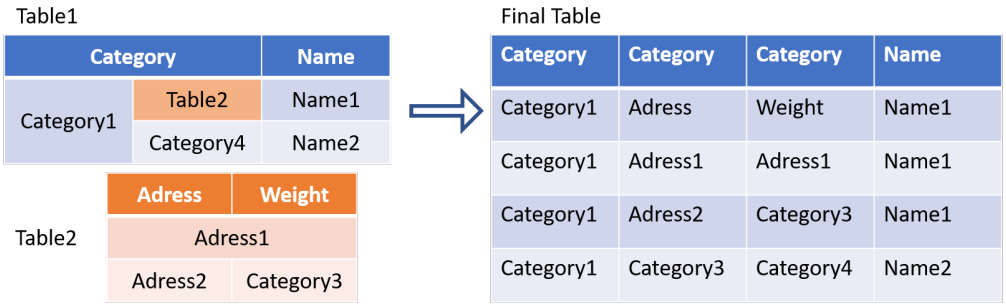

FIGURE 3 – remplissage de la Table1 avec le contenu de la Table2 pour créer la Table finale

4.3 Hétérogénéité décantée

Pour le mélange des alphabets avec des chiffres dans les cellules des tableaux, notre programme lit très bien les cellules et les replace comme il faudraitet écrits correctement. Cependant, les alphabets étrangers sont lus mais sont retournés en chiffrage text. Ceci est une limite de notre programme mais nous entendons que ces traitements peuvent être faits en aval de notre travail, pour un besoin statistique par exemple de nos tableaux.

4.4 Autres problèmes résolus

Outres ces défis qui ont été relevés, nous avions des liens URL qui n'existaient pas ou plus au moment de la compilation. Notre programme prend en compte cette difficulté et passe son chemin

lorsque la page internet n'est pas trouvée.

5 Résultats et statistiques

5.1 Résultats

Nous avons obtenu 1730 Tableaux de 11 types différents que nous avons interprétés comme pertinents. Ils ont été placé dans le chemin /target/wikiCSVs du projet. 33 URLs sur les 330 n'éxistaient pas ou plus lorsque nous les avons cherché, ainsi, le parseur ne pouvant trouver les pages, émet dans la console un message d'erreur avec sa source : `HTTP error fetching URL`. Pour les statistiques qui suivent, nous avons créé un fichier text au chemin /target/statistics.txt. En voici quelques résultats mieux présentés.

5.2 Statistiques sur les tableaux

Pour tous les tableaux récupérés, la moyenne des colonnes et des lignes sont respectivement $9,26$ et $21,3$.

	Minimum	Maximum	Moyenne	Variance
Colonnes	1	90	9,26	50,9
Lignes	1	464	21,3	1001

FIGURE 4 – Statistiques descriptives sur les colonnes et lignes des tableaux récupérés

Parmi les 5286 différents nom de colonnes, les 10 les plus fréquentes sont dans la figure 5 avec leur occurences. On remarque qu'il y a 423 colonnes avec un nom vide ; elles correspondent aux premières colonnes des tableaux à double entrées.

Memory	« »	Fillrate	Trident	Webkit	Gecko	Presto	Model	Licence	Name
574	423	279	229	223	219	219	176	161	155

FIGURE 5 – Noms des 10 colonnes les plus représentées